UNIVERSIDAD CARLOS III DE MADRID

INGENIERÍA SUPERIOR DE TELECOMUNICACIÓN

ESPECIALIDAD EN REDES Y SISTEMAS

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

PROYECTO FIN DE CARRERA

Implementación de un prototipo IEEE 802.21 para realizar handovers entre diferentes tecnologías

Autor: Alberto Martín Ascensión

Tutor: Antonio de la Oliva Delgado

Leganés, Diciembre de 2010

Título: Implementación de un prototipo IEEE 802.21 para realizar handovers entre diferentes tecnologías
Autor: Alberto Martín Ascensión
Director: Antonio de la Oliva Delgado.

EL TRIBUNAL

Presidente: _____

Vocal: _____

Secretario: _____

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día __ de _____ de 2010 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Resumen

La finalidad de este trabajo es la implementación de un prototipo que permite simular la realización de un handover entre dos redes utilizando el protocolo MIH definido en el estándar IEEE 802.21. Dicho prototipo ha sido implementado mediante el lenguaje de programación C.

La implementación del prototipo se lleva a cabo mediante la definición en lenguaje C de las funciones descritas en el estándar IEEE 802.21 necesarias para la optimización de un handover entre dos redes, tanto en el caso de que éste sea iniciado por el Mobile Node como en el caso de que sea iniciado por la red.

Para la simulación del handover se han implementado distintos programas en el lenguaje de programación C que se encargan de realizar los distintos pasos del handover en los elementos presentes en la simulación y que hacen uso de las funciones definidas en este proyecto.

Las pruebas realizadas demuestran que la utilización de los mecanismos definidos en el estándar IEEE 802.21 proporcionan una sustancial mejora en los tiempos empleados para la realización de handovers.

Índice general

Índice de figuras

Índice de tablas

1. Introducción

En la sociedad actual las comunicaciones móviles han adquirido una gran importancia tanto en el ámbito empresarial como en el personal. Hoy en día los usuarios de la redes de telecomunicaciones exigen estar siempre conectados en todo momento a las distintas redes para acceder a todo tipo de servicios.

Las distintas redes existentes proporcionan la cobertura necesaria para que los usuarios tengan siempre un punto de acceso a la red que les permita acceder a los distintos servicios proporcionado por los operadores.

Cuando un usuario ha iniciado una conexión es necesario que ésta se mantenga mientras se desplaza. Puede ocurrir que durante ese desplazamiento el usuario se salga de la zona de cobertura de la red que le da servicio con lo que si antes de que ocurra no se realiza un traspaso (handover) de dicha comunicación ésta se acabaría con el consiguiente perjuicio para el usuario. Para que esto no suceda existen los mecanismos de handover que hacen el traspaso de la comunicación de un punto de acceso que va a dejar de estar disponible a otro que puede seguir proporcionando los recursos para que siga la comunicación.

Los distintos tipo de redes (GSM, UMTS, LTE, Wifi,..) definen mecanismos que llevan a cabo el handover entre los puntos de acceso que pertenecen a la red. Pero entre los distintos tipos de red puede que no esté definido este mecanismo, por ello se ha desarrollado el protocolo MIH dentro del estándar IEEE 802.21 [1] que proporciona los mecanismos para ayudar a la realización del handover entre diferentes tecnologías (handovers verticales). Además los mecanismos definidos en este estándar proporcionan una optimización de los handovers ya definidos en las distintas redes (handovers horizontales) que ayudan a que estos sean más efectivos. También permiten seleccionar la red idónea donde llevar a cabo el handover basándose en las necesidades de la calidad de servicio de la comunicación en curso gracias a la información que obtiene mediante un servicio de información (MIIS) de todas las redes diponibles en un determinado área geográfica.

El objetivo de este proyecto es implementar una parte del estándar IEEE 802.21 mediante el lenguaje de programación C [2] y diseñar un prototipo que utilice las funciones definidas en dicho estándar e implementadas en este proyecto para comprobar la efectividad de los mecanismos de optimización del handover.

El trabajo realizado en este proyecto se presenta en siete capítulos organizados de la siguiente manera:

- En este capítulo se presenta una visión del problema a tratar en este proyecto.

- En el capítulo 2 se realiza una descripción del estándar IEEE 802.21

- En el capítulo 3 se describe la implementación de las distintas funciones y programas implementados en C que permiten realizar el prototipo IEEE 802.21

- En el capítulo 4 se presenta una breve descripción del protocolo Mobile IPv6 [3] utilizado en este proyecto.

● En el capítulo 5 se detalla el entorno experimental creado para realizar las pruebas experimentales que permiten probar las funcionalidades del trabajo desarrollado.

● En el capítulo 6 se muestran los resultados experimentales obtenidos con la utilización de los mecanismos definidos en el estándar IEEE 802.21.

● En el capítulo 7 se presentan las conclusiones obtenidas con la realización de este proyecto.

● En el capítulo 8 se presenta un presupuesto del coste aproximado del proyecto

2. IEEE 802.21: Media Independent Handover Services

2.1 Descripción general

El estándar IEEE 802.21 define mecanismos independientes del método/modo de acceso que posibilitan la optimización del handover entre redes 802 heterogéneas y facilita el handover entre redes 802 y redes celulares.

El propósito es mejorar las prestaciones de los dispositivos móviles facilitando el handover entre redes 802 sean o no de diferentes modos de acceso, incluyendo redes cableadas e inalámbricas, donde el handover no está definido; y hacer posible seamless handovers si la red lo permite.

Este estándar proporciona inteligencia a nivel de enlace e información relacionada con la red a los niveles superiores para la optimización del handover entre redes heterogéneas.

A lo largo del estándar se utiliza el término *media* refiriéndose al método/modo de acceso a un sistema de telecomunicación (esto es, cable, radio, satélite) al contrario de lo que es utilizado en otros aspectos de las comunicaciones (esto es, audio, video).

El estándar está dirigido a dar soporte para el handover tanto a usuarios móviles como a fijos. En usuarios móviles, los handovers se dan cuando cambian las condiciones del enlace radio debido a su movimiento. Para usuarios fijos, los handovers se realizan cuando cambia el entorno de la red, haciendo que una red sea más atractiva que otra.

Este estándar da soporte a otro aspecto importante en la optimización del handover-adaptación del enlace. Ciertas aplicaciones pueden requerir un mayor ancho de banda que el disponible en un determinado momento en el enlace que está siendo utilizado, por lo que se hace necesario adaptar el enlace al ancho de banda necesitado o bien, un handover si dicho enlace no es capaz de soportar los requisitos de la aplicación.

En cualquier caso se debería mantener todo lo posible la continuidad en el servicio durante el handover.

El estándar da soporte para que la información disponible tanto en el dispositivo móvil como en la red pueda ser compartida. Por una parte el dispositivo móvil puede detectar las redes disponibles; por otra la infraestructura de red es capaz de almacenar información general acerca de la red, tal como, lista de celdas vecinas, localización de dispositivos móviles o disponibilidad de servicios de nivel superior. Además tanto los dispositivos móviles como los puntos de acceso a la red, son capaces de soportar múltiples estándares de radio y conexiones simultáneas en más de un interfaz radio.

La red en conjunto puede incluir celdas de diferentes tamaños, de los distintos tipos de red tales como IEEE 802.11 [4], IEEE 802.15, IEEE 802.16, 3GPP y 3GPP2 [5], con solapamiento de la cobertura. El proceso de handover es iniciado

por los reportes de medida y los *triggers* proporcionados por el nivel de enlace. Los reportes de medidas incluyen métricas como calidad de señal, diferencias en los tiempos de sincronización, y tasas de errores de transmisión. De forma más específica, este estándar consta de los siguientes elementos [6]:

- Una estructura que permite la continuidad del servicio cuando un *mobile node* (MN) transita entre dos tecnologías heterogéneas a nivel de enlace. La estructura cuenta con la presencia de un protocolo de gestión de movilidad dentro de los elementos de red que soporta el handover. La estructura presenta modelos de referencia del MIH para diferentes tecnologías a nivel de enlace.

- Un conjunto de funciones que permiten el handover dentro de la pila de protocolos de los elementos de red y una nueva entidad creada en dicha pila llamada MIH Function (MIHF).

- Se define un media independent handover Service Access Point (MIH_SAP) y primitivas asociadas para proporcionar a los usuarios del MIH acceso a los servicios de la MIHF. La MIHF proporciona los siguientes servicios:

  ● El Media Independent Event service que detecta cambios en las propiedades del enlace e inicia los eventos apropiados (triggers) desde los interfaces local y remoto.

  ● El Media Independent Command service que proporciona un conjunto de comandos a los usuarios del MIH para controlar las propiedades del enlace que son relevantes en el handover y en la conmutación entre los enlaces.

  ● El Media Independent Information service que proporciona información de las diferentes redes y sus servicios, permitiendo una decisión más efectiva del handover entre redes heterogéneas.

- La definición de nuevos Service Access Points (SAPs) a nivel de enlace y sus primitivas asociadas para las diferentes tecnologías. Estas nuevas primitivas ayudan a la MIHF a recopilar información y controlar el comportamiento del enlace durante el handover. Los nuevos SAPs se recomiendan como corrección para los estándares de las diferentes tecnologías de enlace existentes.

La Figura 2.1 muestra el lugar de la MIHF dentro de la pila de protocolos de un Mobile Node o entidad de red con varios interfaces. La MIHF proporciona servicios a los usuarios del MIH a través de un solo interfaz (el MIH service access point) y obtiene servicios de los niveles inferiores a través de una variedad de interfaces (media-specific SAPs)

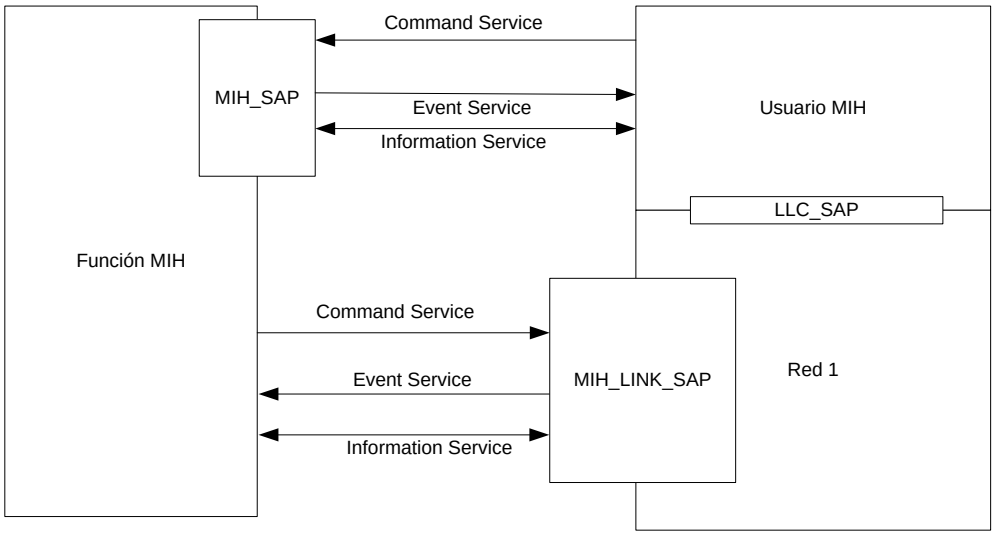

Figura 2.1: Servicios MIH

Para el desarrollo del estándar se han tenido en cuenta los siguientes supuestos:

- El MN es capaz de soportar múltiples tecnologías de enlace, como inalámbricas, cableadas o una mezcla de ambas.

- La MIHF es una entidad lógica, cuya definición es independiente de si está en el MN o en la red.

- La MIHF, sin tener en cuenta si está localizada en el MN o en la red, recibe y transmite información sobre la configuración y condiciones de las redes de acceso alrededor del MN. Esta información se origina en los diferentes niveles de la pila de protocolos del MN o en varios elementos de la red:

  ● Cuando la información es originada en un elemento remoto de la red, la MIHF de la entidad local la obtiene por medio del intercambio de mensajes con la instancia MIHF de su par que reside en el elemento remoto.

  ● Cuando la información se origina en los niveles inferiores de la pila de protocolos del MN o de la entidad de red, la MIHF de la entidad la obtiene localmente a través de las primitivas de los SAPs que define el interfaz de la MIHF con los niveles inferiores.

La intención del estándar es proporcionar una inteligencia genérica a nivel de enlace independiente de los dispositivos móviles o de las redes radio. Este estándar tiene la intención de proporcionar un interfaz genérico entre los usuarios del nivel de enlace de la pila de protocolos de gestión de la movilidad y los niveles de enlace específicos del método de acceso existentes que definen el 3GPP, 3GPP2 y los estándares de la familia 802.

2.2 Definiciones

- *Punto de conexión candidato (candidate point of attachment (candidate PoA)*: Un Punto de conexión (PoA) que se evalúa para una posible conmutación del enlace.

- *Handover*: Proceso por el cual un nodo móvil obtiene recursos y preserva el tráfico cuando ocurre un evento de conmutación del enlace. Los mecanismos y niveles del protocolo involucrados en el handover pueden variar con el tipo de evento de conmutación del enlace. Existen diferentes tipos de handover que se definen en base a cómo los recursos son preservados para el mantenimiento del tráfico: hard handover, soft handover y seamless handover.

- *Hard Handover*: Handover donde los recursos para mantener el tráfico no están disponibles desde la interrupción de los mismos en el enlace fuente hasta la restauración de dichos recursos en el enlace destino (break-before-make).

- *Horizontal handovers*: Un handover donde el nodo móvil se mueve entre puntos de conexión del mismo tipo de enlace.

- *Entidad de red MIH*: Entidad de red con capacidades MIHF

- *Punto de servicio MIH (MIH PoS)*: Instancia MIHF de la red que intercambia mensajes MIH con una MIHF de un MN. La misma entidad de red MIH incluye un MIH PoS para cada MN (que tenga habilitado el protocolo MIH) con el cual intercambia mensajes MIH. Un solo MIH PoS puede prestar más un servicio MIH. La misma entidad de red puede tener múltiples MIH PoS que proporcionan diferentes servicios MIH a los nodos móviles. Para una entidad de red que tiene varios interfaces la noción de MIH Pos está asociada a la entidad en sí misma y no sólo con uno de los interfaces.

- *Nodo MIH*: Entidad con capacidad MIHF (MN o red).

- *MIH non PoS*: Una entidad de red MIH que puede intercambiar mensajes directamente con otras entidades de red MIH pero no puede intercambiar mensajes directamente con nodos móviles MIH.

- *Protocolo de transporte MIH*: Protocolo para el transporte de mensajes MIH entre entidades MIH.

- *Usuarios MIH*: Entidades que usan los servicios proporcionados por la MIHF. Usan los MIH_SAP para interactuar con la MIHF.

- *Media independent handover function (MIHF)*: Función que está cargo de los servicios MIH.

- *Transacción MIH*: La combinación de un mensaje MIH de petición y un mensaje MIH de respuesta, una indicación MIH, o un mensaje MIH de respuesta y cualquier mensaje de asentimiento asociado.

- *Nodo móvil (Mobile Node, MN)*: Nodo de comunicación que puede cambiar su punto de conexión de un enlace a otro.

- *Punto de conexión de red (network point of attachment, network PoA o PoA)*: Punto de terminación de red a nivel de enlace que incluye un nodo móvil como el otro punto de terminación.

- *Seamless handover*: Handover asociado con un cambio de enlace entre puntos de conexión, donde el nodo móvil o no experimenta degradación en la calidad de servicio, seguridad y recursos, o la degradación en los parámetros de servicio es aceptable.

- *Serving network*: Una red que proporciona servicios al usuario.

- *PoA en servicio (serving PoA)*: El PoA del enlace actual que está siendo usado por el nodo móvil.

- *Soft handover*: Handover donde los recursos para el tráfico están siempre disponibles mientras la conexión del nodo móvil se transfiere de un punto de conexión al otro. La red asigna recursos de transporte al punto de conexión destino antes de que ocurra la conmutación del enlace (make-before-break).

- *Punto de conexión objetivo (target point of attachment, target PoA)*: Un PoA candidato que ha sido seleccionado para convertirse en el nuevo PoA en servicio.

- *Handovers verticales*: Un handover donde el nodo móvil se mueve entre PoA de diferentes tipos de enlace.

2. 3 Arquitectura general

2.3.1 Introducción

Este estándar soporta los diferentes métodos de handover, es decir, hard y soft. Los diferentes factores que afectan al handover y que son tratados en el estándar son los siguientes:

- Continuidad del servicio: Es definida como la continuidad en el servicio durante y después del handover sin la intervención del usuario. Puede haber un cambio en la calidad del servicio como consecuencia de la transmisión entre diferentes redes debido a la variación de capacidades y características de las redes de acceso. Este estándar especifica elementos esenciales que permiten la continuidad del servicio.

- Calidad de servicio: La calidad de servicio (QoS) que experimenta una aplicación depende de la exactitud, velocidad, y disponibilidad de la información transferida en el canal de comunicación. Este estándar proporciona soporte para una completa aplicación de los requerimientos de QoS durante el handover. Hay dos aspectos de QoS a considerar. El primero, es la calidad de servicio experimentada durante el handover. El segundo es la QoS considerada como parte de la decisión del handover. El estándar incluye mecanismos que soportan ambos aspectos de la QoS.

- Descubrimiento de redes: Este estándar define la información que ayuda al descubrimiento de redes y especifica el medio por el cual se obtiene dicha información y es puesta a disposición del usuario.

- Selección de red: Es el proceso por el cual el MN u otra entidad de la red selecciona una red para establecer una conexión a nivel de red. Este estándar especifica los medios por los cuales dicha información es puesta a disposición de los usuarios MIH para que sea posible una efectiva selección de la red.

- Gestión de potencia: Este estándar permite que el MN descubra diferentes tipos de redes inalámbricas evitando el encendido de múltiples dispositivos de radio y un excesivo escaneo de los radioenlaces. Con esto, se minimiza la potencia consumida por los dispositivos móviles en la búsqueda de potenciales redes para llevar a cabo el handover.

-. Política de handover: El rol de la MIHF es facilitar los handovers y proporcionar información a la entidad encargada de seleccionar la red. La MIHF asiste al elemento encargado de seleccionar la red mediante la ayuda proporcionada por el Event Service, el Command Service y el Information Service.

2.3.2 Principios generales de diseño

Este estándar se basa el los siguientes principios generales de diseño:

1.- La MIHF es una entidad lógica que facilita la toma de decisión del handover. Los usuarios del MIH toman la decisión del handover basándose en las salidas proporcionadas por la MIHF.

2.- La MIHF servicios a los niveles superiores. El servicio de primitivas definido por este interfaz está basado en entidades de protocolo específicas para cada tecnología de las diferentes redes de acceso.

3.- Los protocolos de gestión de la movilidad de nivel superior se encargan de definir los mecanismos de señalización para los handovers verticales. Adicionalmente, las diferentes tecnologías de acceso tienen definidos mecanismos de señalización para los handovers horizontales. El propósito de este estándar es facilitar el handover y maximizar la eficiencia del mismo proporcionando información de tanto de enlace como de red.

4.- El estándar proporciona soporta para eventos remotos.

5.- Este estándar ofrece una operación transparente con los equipos existentes.

2.3.3 Servicios proporcionados por la MIHF: Visión general

Este estándar define servicios que facilitan los handovers entre enlaces de redes heterogéneas.

- *Media Independent Event Service (MIES) [7]*: Proporciona clasificación, filtrado, y reportes correspondientes a cambios dinámicos en las características, estado y calidad del enlace.

- *Media Independent Command Service (MICS)*: Permite a los usuarios MIH gestionar y controlar el comportamiento del enlace en relación al handover y a la movilidad.

- *Media Information Service*: Proporciona detalles de las características y servicios proporcionados por la red en servicio y las redes vecinas. La información proporciona un mejor acceso al sistema y handovers más eficientes.

La MIHF proporciona servicios síncronos y asíncronos a través de los SAPs para los niveles de enlace y los usuarios MIH. En el caso de sistemas con varios interfaces de red, los usuarios MIH usan el Event Service, el Command Service y el Information Service proporcionados por la MIHF para gestionar, determinar y controlar el estado de los interfaces de nivel inferior.

Estos servicios proporcionados por la MIHF ayudan a los usuarios MIH en el mantenimiento de la continuidad del servicio, adaptación de la calidad de servicio, conservación de la batería, descubrimiento de redes, y selección de enlace. En un sistema que tiene interfaces de red heterogéneos de tipos IEEE 802 y celular (3GPP, 3GPP2), la MIHF ayuda a los usuarios MIH a implementar procedimientos eficaces para asociar servicios entre interfaces de redes heterogéneas. Los usuarios MIH utilizan los servicios proporcionados por la MIHF para realizar peticiones de los recursos requeridos en el handover entre redes heterogéneas.

Los servicios MIH en los nodos móviles facilitan los seamless handover entre redes heterogéneas. Son usados por los usuarios MIHF como son los protocolos de gestión de movilidad, por ejemplo, MobileIP.

2.3.3.1 Media Independent Event Service

Los eventos indican cambios en el estado y en la transmisión a nivel físico y de enlace o predice cambios en estos niveles. El Event Service se utiliza también para indicar acciones de gestión o estados de comandos en la parte de red o en algunas entidades de gestión.

Los eventos son originados por la MIHF (eventos MIH) o en cualquier nivel inferior (eventos de enlace) dentro de la pila de protocolos de un MN o nodo de red.

El destino de un evento es la MIHF o alguna entidad de nivel superior. El receptor del evento puede estar dentro del nodo que origina el evento o en un nodo remoto. El destino de un evento se establece mediante un mecanismo de suscripción que permite a un nodo suscribirse a los distintos tipos de eventos según sus necesidades.

En el caso de eventos locales, los mensajes se propagan desde los niveles inferiores a la MIHF y desde la MIHF a los niveles superiores. En el caso de eventos remotos, los mensajes se propagan desde la MIHF de la pila de protocolos hasta la MIHF de la pila de protocolos de su par.

El Event Service se usa para detectar la necesidad de realizar un handover. Puede, además, llevar adicionalmente datos sobre el contexto tales como identificadores de nivel 2 y nivel 3.

2.3.3.2 Media Independent Command Service

El Command Service permite a los niveles superiores controlar la capa física y de enlace. Las capas superiores controlan la reconfiguración y la selección de un enlace apropiado a través de un conjunto de comandos del handover. Si la MIHF soporta el Command Service todos los comandos MIH son obligatorios. Cuando una MIHF recibe un comando, siempre se espera que éste se ejecute.

Los comandos son invocados por los usuarios MIH (Comandos MIH) o por la misma MIHF (Comandos de Enlace).

El destino de un comando es la MIHF o algún nivel inferior. El receptor del comando está localizado en la pila de protocolos que originó el comando, o dentro de la pila de protocolos remota.

En el caso de comandos locales, los mensajes se propagan normalmente desde los usuarios MIH a la MIHF y luego desde la MIHF a las capas inferiores. En el caso de comandos remotos, los mensajes se propagan desde los usuarios MIH por medio de la MIHF en la pila de protocolos a su par MIHF de la pila de protocolos.

Los comandos generalmente transportan las decisiones de las capas superiores a las inferiores en el dispositivo local o a una entidad remota.

Este estándar facilita tanto los handovers iniciados por el nodo móvil como por la red.

El estándar soporta un conjunto de comandos independientes del modo de acceso que ayudan a la selección de la red bajo diferentes condiciones. Estos comandos permiten tanto al MN como a la red iniciar handovers e intercambiar información sobre las redes disponibles y negociar la mejor red bajo las condiciones dadas.

2.3.3.3 Media Independent Information Service

El Media Independent Information Service (MIIS) proporciona la estructura y los mecanismos adecuados para que una entidad MIHF pueda descubrir y obtener información de las redes que existen en un área geográfica que faciliten el handover.

El MIIS proporciona un conjunto de elementos de información (IEs), la estructura de la información y su representación, y un mecanismo de petición/respuesta

para la transmisión de la información. Dicha información está disponible en un servidor desde donde la MIHF del MN puede obtenerla.

La información está disponible por medio de las capas inferiores y superiores. A nivel 2 a través de un puerto (seguro o no). Donde no se puede acceder a la información a través del nivel 2 o a la información a este nivel no es suficiente para tomar una decisión correcta sobre el handover, dicha información se puede obtener por medio de los niveles superiores. Este estándar permite tanto el transporte a nivel 2 como al 3 para el acceso a la información.

El MIIS proporciona parámetros estáticos a nivel de enlace tales como información del canal, la dirección MAC e información de seguridad de un PoA.

La información proporcionada por el MIIS se ajusta a la estructura y la semántica especificadas en el estándar. El MIIS especifica una manera común de representar la información para las diferentes tecnologías usando un formato estandarizado como el XML o por medio de la codificación binaria.

El MIIS define un mecanismo unificado para las entidades de nivel superior que proporciona información de la red candidata al handover donde las redes son heterogéneas en una localización geográfica determinada.

2.3.4 MIHF: Modelo de comunicación

Las MIHF comunican unas con otras con varios objetivos. El MN intercambia información MIH con su MIH PoS. La MIHF en una entidad de red se convierte en un MIH PoS cuando comunica directamente con una MIHF de un MN. Cuando una MIHF en una entidad de red no tiene una conexión directa con el MN, entonces no actúa como un MIH PoS para ese MN en particular. Sin embargo, la misma entidad de red MIH puede actuar como MIH PoS para un MN distinto.

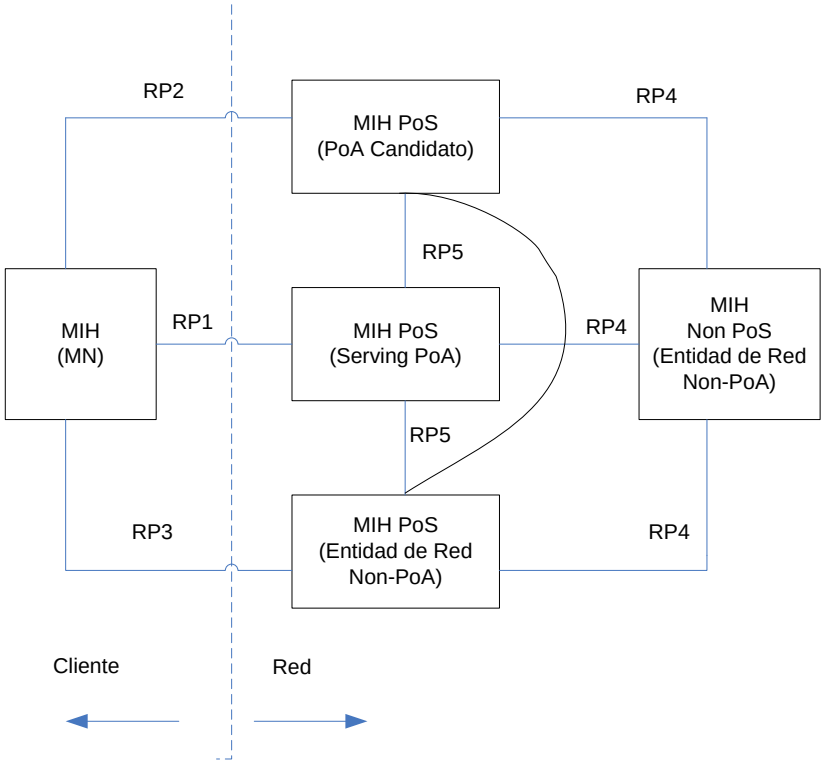

Figura 2.2: Modelo de comunicación MIH

El modelo de comunicación asigna diferentes roles a la MIHF (Figura 2.2) dependiendo de su posición en el sistema, estos roles son los siguientes:

- MIHF en el MN
- MIH PoS en la entidad de red que incluye el PoA en servicio del MN
- MIH PoS en la entidad de red que incluye un PoA candidato para el MN
- MIH PoS en una entidad de red que no incluye un PoA para el MN
- MIH non-PoS en una entidad de red que no incluye un PoA para el MN

Este modelo identifica puntos de referencia entre diferentes instancias de las MIHF. Estos puntos de referencia (Figura 2.2) son los siguientes:

- Punto de referencia 1 (RP1): Procedimientos de la MIHF entre la MIHF en el MN y el MIH PoS en la entidad de red de su PoA en servicio. RP1 abarca los interfaces de comunicación tanto a nivel 2 como 3 y superiores. El contenido de la MIHF pasado sobre RP1 se asocia al MIIS, MIES o MICS.

- Punto de referencia 2 (RP2): Procedimientos de la MIHF entre la MIHF en el MN y el MIH PoS en la entidad de red de un PoA candidato. RP2 abarca los interfaces de comunicación tanto a nivel 2 como 3 y superiores. El contenido de la MIHF pasado sobre RP2 se asocia al MIIS, MIES o MICS.

- Punto de referencia 3 (RP3): Procedimientos de la MIHF entre la MIHF en el MN y el MIH PoS de un non-PoA de la entidad de red. RP3 abarca los interfaces de comunicación sobre nivel 3 y posiblemente protocolos de transporte de nivel 2 como puentes Ethernet o MPLS. El contenido de la MIHF pasado sobre RP3 se asocia al MIIS, MIES o MICS.

- Punto de referencia 4 (RP4): Procedimientos de la MIHF entre un MIH PoS en una entidad de red y una instancia MIH non-PoS en otra entidad de red. RP4 abarca los interfaces de comunicación sobre nivel 3 y superiores. El contenido de la MIHF pasado sobre RP4 se asocia al MIIS, MIES o MICS.

- Punto de referencia 5 (RP5): Procedimientos de la MIHF entre dos instancias MIH PoS en diferentes entidades de red. RP5 abarca los interfaces de comunicación sobre nivel 3 y superiores. El contenido de la MIHF pasado sobre RP5 se asocia al MIIS, MIES o MICS.

2.3.5 Modelos de referencia MIH para tecnologías a nivel de enlace

La MIHF proporciona servicios síncronos y asíncronos a través de Service Access Points para los usuarios MIH.

La MIHF es una entidad de gestión que obtiene información del nivel de enlace desde los niveles inferiores de la pila de protocolos y desde otros nodos remotos, y coordina la decisión de hacer un handover con su par MIHF de la red.

El protocolo MIH proporciona la capacidad de transferir mensajes MIH entre entidades MIHFs pareadas tanto a nivel 2 como a nivel 3. Estos mensajes transmiten información sobre las diferentes redes disponibles y también proporcionan las capacidades para el handover y la conmutación entre redes.

2.3.5.1 Modelo de referencia MIH y SAPs

El modelo de referencia MIH (Figura 2.3) incluye los siguientes SAPs:

- MIH_SAP: Interfaz independiente del modo de acceso de la MIHF con las capas superiores de la pila de protocolos.

- MIH_LINK_SAP: Interfaz abstracto dependiente del modo de acceso de la MIHF con las capas inferiores de la pila de protocolos específicos del medio de acceso.

- MIH_NET_SAP: Interfaz abstracto dependiente del modo de acceso que proporciona servicios de transporte sobre el plano de datos en el nodo local, dando soporte al intercambio de información MIH y mensajes con la MIHF remota. Para todos los servicios de transporte sobre nivel 2, el MIH_NET_SAP usa las primitivas especificadas por el MIH_LINK_SAP.

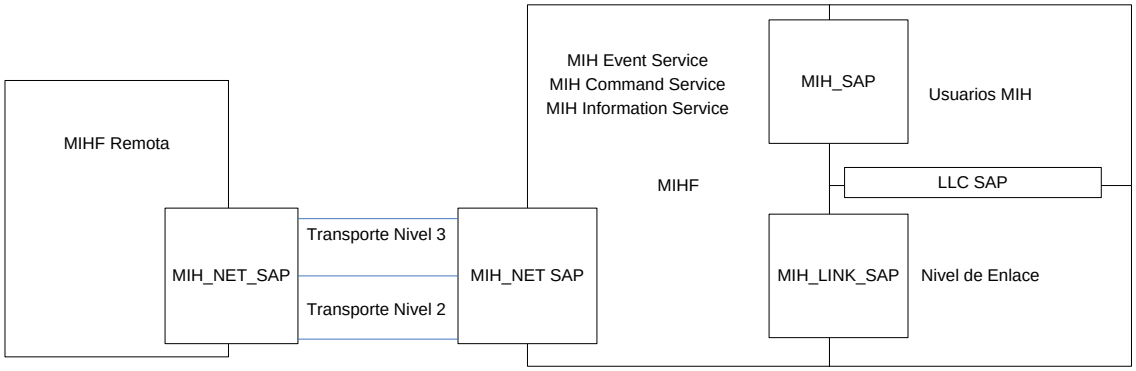

Figura 2.3: Modelo de referencia MIH y SAPs

En los modelos de referencia específicos de cada método de acceso, el MIH_SAP siempre mantiene el mismo nombre y el mismo conjunto de primitivas. El SAP dependiente del modo de acceso (el cual es una instanciación del MIH_LINK_SAP específico de la tecnología), adopta nombres específicos del método de acceso y conjuntos de primitivas, normalmente reutilizando nombres y primitivas que ya existen en SAPs de los niveles inferiores. Las primitivas definidas en el MIH_LINK_SAP dan lugar a correcciones de los SAPs específicos del modo de acceso debido a que se añaden funcionalidades que son definidas para conectar con la MIHF. Todas las comunicaciones de la MIHF con las capas inferiores de la pila de protocolos específicas de cada método de acceso se hacen por medio de instanciaciones del MIH_LINK_SAP específicos de cada modo de acceso.

El intercambio de mensajes entre pares de instancias MIHF es sensible a varios factores, tales como la naturaleza de los nodos de la red que contienen el par MIHF, la naturaleza de la red de acceso y la disponibilidad de las capacidades MIH en el PoA.

Las relaciones que pueden existir entre la MIHF y otros componentes se representan en la siguiente figura (Figura 2.4):

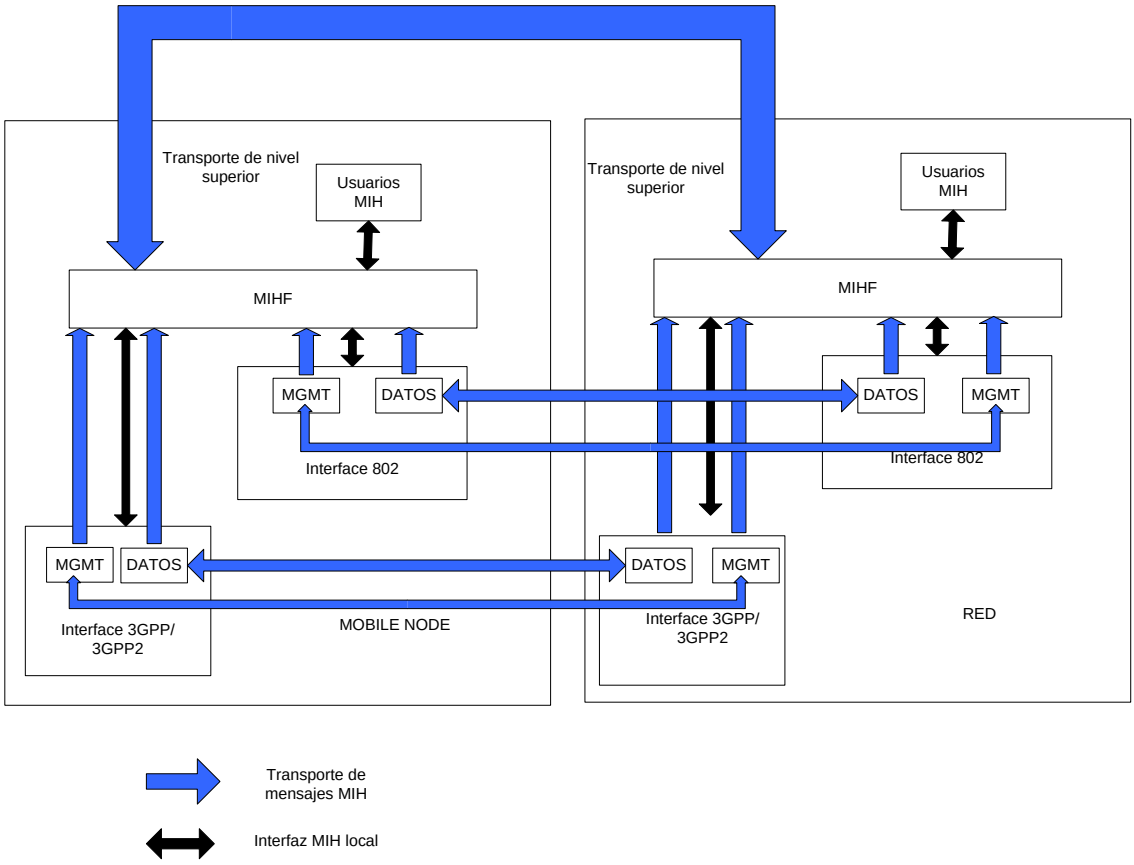

Figura 2.4: Tipos de relación de las MIHF

2.3.6 Service Access Points (SAPs)

La MIHF se comunica con otras capas y planos funcionales por usando los Service Access Points (SAPs). Cada SAP consta de un conjunto de primitivas que especifican como interactúan el usuario del servicio y el proveedor del mismo.

La especificación de la MIHF incluye la definición de SAPs que son independientes del modo de acceso y recomendaciones para definir o extender otros SAPs que dependen del modo de acceso. Los SAPs independientes del método de acceso permiten a la MIHF proporcionar servicios a las capas superiores de la pila de protocolos que gestionan la movilidad, al plano de gestión de red y al plano de datos. El MIH_SAP y sus primitivas asociadas proporcionan el interfaz desde la MIHF a los niveles superiores de la pila de protocolos de gestión de la movilidad. Las capas superiores necesitan suscribirse a la MIHF como usuarios para recibir eventos generados por la MIHF y también para los eventos del nivel de enlace que se originan en las capas inferiores a la MIHF pero que son pasados a los usuarios MIH a través de la MIHF. Los usuarios MIH envían directamente comandos a la MIHF local usando el servicio de primitivas del MIH_SAP. La comunicación entre dos MIHFs se lleva a cabo mediante mensajes del protocolo MIH.

Los SAPs que dependen del método de acceso permiten a la MIHF usar servicios de las capas inferiores de la pila de protocolos de gestión de movilidad y sus planos de gestión. Todas las entradas (incluyendo los eventos) de los niveles inferiores a la MIHF son proporcionados a través de los SAPs específicos del modo de acceso tales como MAC SAPs, PHY SAPs y LLC SAps. Los comandos de enlace generados por la MIHF para controlar los niveles MAC y físico (PHY) durante el handover forman parte de los SAPs MAC/PHY específicos del modo de acceso y ya están definidos en otros lugares.

La relación entre los SAPs y la MIHF para diferentes redes son (Figura 2.5):

- MIH_SAP: especifica un interfaz independiente del modo de acceso entre la MIHF y las capas superiores de la pila de protocolos de gestión de la movilidad. Las capas superiores necesitan subscribirse a la MIHF como usuarios para recibir eventos generados por la MIHF y también para los eventos de nivel de enlace que se originan en las capas inferiores pero que son pasados a los usuarios de la MIHF a través de la MIHF. Los usuarios de la MIHF envían comandos directamente a la MIHF local usando el servicio de primitivas del MIH_SAP.

- MIH_LINK_SAP: especifica un interfaz abstracto dependiente del modo de acceso entre la MIHF y as capas inferiores de la pila de protocolos específica de cada método de acceso. Para diferentes tecnologías a nivel de enlace los SAPs específicos del método de acceso proporcionan la funcionalidad del MIH_LINK_SAP.

- MIH_NET_SAP: especifica un interfaz abstracto dependiente del modo de acceso de la MIHF que proporciona servicios de transporte sobre el plano de datos en el nodo local, dando soporte al intercambio de mensajes MIH e información con MIHFs remotas.

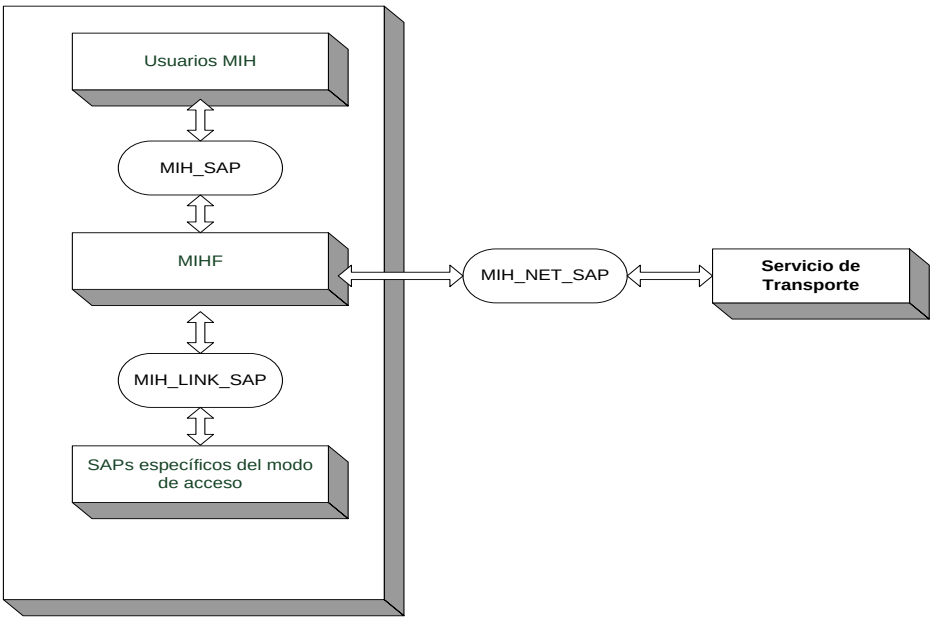

Figura 2.5: Relación entre diferentes MIHF SAPs

2.3.6.1 SAPs dependientes del método de acceso

Cada tecnología de nivel de enlace especifica sus propios SAPs dependientes de la tecnología. Para cada tecnología de nivel de enlace, el MIH_LINK_SAP se mapea a los SAPs de dicha tecnología.

● MIH_LINK_SAP: Este SAP define un interfaz abstracto dependiente del modo de acceso entre la MIHF y las diferentes tecnologías a nivel de enlace.

● MIH_NET_SAP: Define un interfaz de la MIHF abstracto dependiente del modo de acceso que proporciona servicios de transporte sobre el plano de datos en el nodo local, dando soporte al intercambio mensajes MIH e información con las MIHFs remotas. Para el nivel 2 este SAP usa las primitivas proporcionadas por el MIH_LINK_SAP.

● MLME_SAP: Define el interfaz entra la MIHF y el plano de gestión de una red 802.11. Se usa para el envío de mensajes MIH entre la MIHF y las entidades locales de nivel de enlace, como también entre entidades MIHF asociadas.

● MSGCF_SAP: Definido en el IEEEP802.11u/D3.0, proporciona servicios a la MIHF basados en las maquinas de estado del IEEE 802.11 MAC e interactúa entre las distintas subcapas IEEE 802.11.

● LSAP: Definido en el IEEE Std 802.2, proporciona el interfaz entre la MIHF y el subnivel LLC en redes IEEE 802.3 e IEEE 802.11. Es usado para intercambios MIH locales entre la MIHF y los niveles inferiores y para el transporte a nivel de enlace de mensajes entre enlaces de acceso IEEE 802

2.3.6.2 SAP independiente del método de acceso: MIH_SAP

El MIH_SAP define el interfaz independiente del método de acceso entre la MIHF y los usuarios MIH tales como los protocolos de movilidad de nivel superior o una función de handover que podría residir en las capas superiores o una entidad de transporta de los niveles superiores.

2.3.7 Protocolo MIH

El protocolo MIH define el formato de mensajes (cabecera y datos) que son intercambiados entre entidades MIHF remotas y los mecanismos independientes del modo de acceso que dan soporte a la entrega del mensaje.

Todas las PDUs del protocolo MIH se deben definir usando el valor 8917 en el tipo Ethernet.

Los mensajes del protocolo MIH se envían en el plano de datos usando un mecanismo de transporte apropiado a nivel 2 y 3. A nivel 3 se usa TCP, UDP o SCTP sobre IP. A nivel 2 se usa el tipo Ethernet con valor 8917.

2.4 Servicios MIHF

La MIHF proporciona el Media Independent Event Service, el Media Indepedent Command Service y el Media Information Service que facilitan los handovers entre redes heterogéneas. Estos servicios son gestionados y configurados a través de las primitivas de servicio de gestión.

2.4.1 Servicio de Gestión

Anteriormente a la provisión de servicios de una MIHF a otra, las entidades MIH necesitan ser configuradas apropiadamente. Esto se hace a través de las siguientes funciones de gestión de servicios:

- MIH Capability Discovery
- MIH Registration
- MIH Event Subscription

Para saber los servicios que son soportados por una MIHF asociada, el nodo MIH ejecuta las funciones MIH Capability Discovery. Dicho nodo ejecuta estas funciones con diferentes MIHFs asociadas para decidir con cual registrarse.

2.4.1.1 Primitivas de Servicio de Gestión

La siguiente tabla (Tabla 2.1) muestra el conjunto de primitivas del servicio de gestión.

Primitiva	**(L)ocal, (R)emoto**	**Descripción**
MIH_Capability_Discover	L, R	Descubre las capacidades de una MIHF local o remota
MIH_Register	R	Registro con una MIHF remota
MIH_DeRegister	R	Desregistro de una MIHF remota
MIH_Event_Subscribe	L, R	Suscripción para uno o más eventos MIH con una MIHF local o remota
MIH_Event_Unsubscribe	L, R	Dar de baja la suscripción para uno o más eventos MIH con una MIHF local o remota

L (Local): puede ser invocada por un usuario MIH local
R (Remoto): puede ser invocada por un usuario MIH remoto

Tabla 2.1: Primitivas del servicio de gestión

2.4.1.2 MIH Capability Discovery

Este procedimiento es usado por un usuario MIH para averiguar las capacidades de las MIHFs locales o remotas en términos de servicios MIH.

2.4.1.3 MIH Registration

Proporciona un mecanismo para que una entidad MIH indique su presencia a una entidad MIH asociada.

2.4.1.4 MIH Event Subscription

Este mecanismo permite a los usuarios MIH suscribirse a un determinado conjunto de eventos originados en una MIHF local o remota.

2.4.1.5 Network Comunication

Esta función proporciona servicios de transporte en el plano de datos en el nodo local, dando soporte al intercambio de información y mensajes MIH entre la MIHF local y remota. Para transporte sobre nivel 2, MIH_NET_SAP utiliza las primitivas especificadas por el MIH_LINK_SAP. Para transporte sobre nivel 3 las primitivas son especificadas por el MIH_NET_SAP.

2.4.2 Media Independent Event Service

Los handovers pueden ser iniciados por el MN o por la red. Los eventos relevantes al handover se originan en PHY, MAC o la MIHF en el MN, en el PoA de la red, o en el PoS. Por lo tanto, la fuente de estos eventos puede ser local o remota. Se necesita un protocolo de transporte para dar soporte a estos eventos. La seguridad también es importante en estos protocolos de transporte.

Varias entidades de los niveles superiores pueden estar interesadas en estos eventos a la vez. Así, estos eventos pueden tener varios destinatarios. Las entidades de las capas superiores se suscriben para recibir notificaciones de eventos de una fuente en particular. La MIHF ayuda en el envío de estos eventos a los distintos destinatarios.

Los eventos son tratados como eventos discretos. La notificación de los mismos se genera de forma asíncrona. Entonces, todos los usuarios MIH que quieren recibir notificaciones de eventos necesitan suscribirse a unos eventos en particular.

El Event Service se divide en dos categorías, Eventos de Enlace (Link Events) y Eventos MIH (MIH Events) (Figura 2.6). Ambos van desde los niveles inferiores a los superiores. Los Link Events se definen como eventos que se originan en las entidades fuente de eventos inferiores a la MIHF y terminan en la MIHF. Los MIH Events se definen como eventos que se originan en la MIHF, o son Link Events propagados por la MIHF a los usuarios MIH.

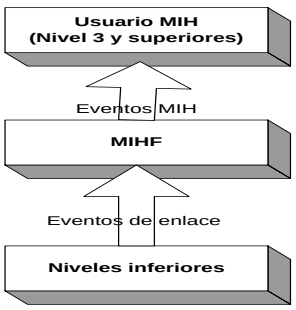

Figura 2.6: Eventos MIH y eventos de enlace

Un evento puede ser local o remoto; un evento local es el que se propaga entre diferentes niveles dentro de la pila de protocolos local de una entidad MIH, mientras que un evento remoto (Figura 2.7) es el que atraviesa la red de una entidad MIH a otra entidad MIH.

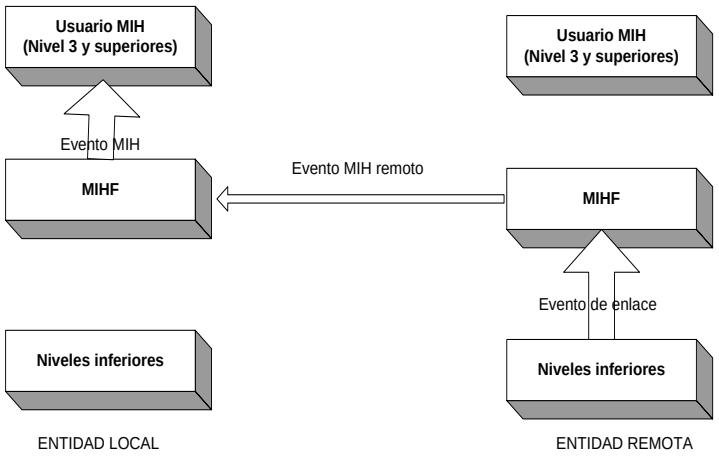

Figura 2.7: Eventos MIH remotos

2.4.2.1 Suscripción a Eventos

La Suscripción a Eventos (Event Subscription) proporciona un mecanismo a las entidades de los niveles superiores para recibir eventos de manera selectiva. La Suscripción a Eventos se puede dividir en Suscripción a Eventos de Enlace (Link Event Subscription) y Suscripción a Eventos MIH (MIH Event Subscription). La Suscripción a Eventos de Enlace es realizada por la MIHF con las entidades fuente de los eventos para determinar qué eventos de cada fuente es capaz de proporcionar. La Suscripción a Eventos MIH se realiza por las entidades de nivel superior con la MIHF para seleccionar los eventos que se quieren recibir.

2.4.2.2 Eventos de Enlace

El Media Independent Event Service soporta las siguientes categorías de Eventos de Enlace:

- Eventos de cambio de estado de los niveles físico (PHY) y MAC: Estos eventos se corresponden con los cambios en el estado de los niveles físico y MAC.

- Eventos de parámetros de enlace: Estos eventos se deben a los cambios en los parámetros del nivel de enlace.

- Eventos predictivos: Estos eventos transmiten un posible cambio en las condiciones del enlace en un futuro cercano basado en predicciones pasadas y presentes.

- Eventos de handover de enlace: Estos eventos informan a los niveles superiores sobre la ocurrencia de un handover/conmutación del enlace a nivel 2.

- Eventos de transmisión de enlace: Estos eventos indican el estado de la transmisión a nivel de enlace de las PDUs de los niveles superiores. Esta información es usada por los niveles superiores para una mejor gestión de los buffers minimizando la pérdida de datos debido al handover.

Evento	Tipo de Evento	Descripción
Link_Detected	Cambio de estado	Se ha detectado un enlace de una nueva red de acceso. Normalmente se genera en el MN cuando se detecta el primer PoA de una red. Este evento no se genera cuando se descubren los siguientes PoA de la misma red
Link_Up	Cambio de estado	La conexión a nivel 2 se ha establecido y el enlace está disponible para su uso. Es un evento discreto
Link_Down	Cambio de estado	La conexión a nivel 2 se ha perdido y el enlace no está disponible para su uso. Es un evento discreto
Link_Parameters_Report	Parámetros de enlace	Los parámetros del enlace han cruzado el umbral predefinido
Link_Going_Down	Predictivo	Las condiciones del enlace se han degradado y la pérdida de conexión es inminente
Link_Handover_Inminent	Handover de enlace	El handover a nivel 2 es inminente basado en cambios del enlace
Link_Handover_Complete	Handover de enlace	El handover de nivel 2 a un nuevo PoA ha sido completado
Link_PDU_Transmit_Status	Transmisión de enlace	Estado de transmisión de una PDU

Tabla 2.2: Eventos de Enlace

2.4.2.3 Eventos MIH

En la siguiente tabla (Tabla 2.3) se describen los Eventos MIH

Evento	(L)ocal (R)emoto	Descripción
MIH_Link_Detected	L,R	Se ha detectado un enlace de una nueva red de acceso. Normalmente se genera en el MN cuando se detecta el primer PoA de una red. Este evento no se genera cuando se descubren los siguientes PoA de la misma red
MIH_Link_Up	L,R	La conexión a nivel 2 se ha establecido y el enlace está disponible para su uso.
MIH_Link_Down	L,R	La conexión a nivel 2 se ha perdido y el enlace no está disponible para su uso.
MIH_Link_Parameters_Report	L,R	Los parámetros del enlace han cruzado el umbral predefinido y necesitan ser reportados
MIH_Link_Going_Down	L,R	Las condiciones del enlace se han degradado y la pérdida de conexión es inminente
MIH_Link_Handover_Inminent	L,R	El handover a nivel 2 es inminente basado en cambios del enlace o en la información adicional disponible en la red
MIH_Link_Handover_Complete	L,R	El handover de nivel 2 a un nuevo PoA ha sido completado
MIH_Link_PDU_Transmit_Status	L	Estado de transmisión de una PDU

Tabla 2.3: Eventos MIH

2.4.3 Media Independent Command Service

El Media Independent Command Service (MICS) hace referencia a los comandos enviados por los usuarios MIH a los niveles inferiores en el modelo de referencia. Los usuarios MIH utilizan el servicio de comandos para determinar el estado de enlaces y/o controlar el dispositivo para un rendimiento óptimo. También permite a los usuarios MIH optimizar las políticas de handover.

La información proporcionada por el MICS es dinámica y está formada por los parámetros del enlace tales como la intensidad de la señal y la velocidad del enlace, mientras que la información proporcionada por el MIIS es menos dinámica o estática y está formada por parámetros como los operadores de la red e información de las capas superiores.

Los comandos se clasifican en dos categorías: Comandos MIH (MIH Commands) y Comandos de Enlace (Link Commands) (Figura 2.8).

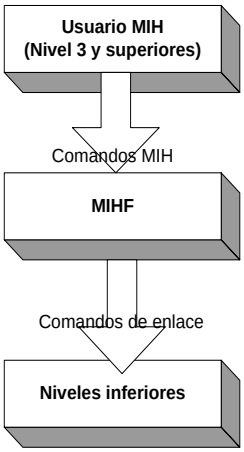

Figura 2.8: Comandos MIH y comandos de enlace

Los Link Commands son originados por la MIHF y dirigidos a los niveles inferiores. Estos comandos controlan principalmente el comportamiento de las entidades de los niveles inferiores. Los Link Commands son solamente locales.

Los MIH Commands son generados por los usuarios MIH y enviados a la MIHF. Pueden ser locales o remotos. Los MIH Commands locales son enviados por los usuarios MIH a la MIHF de la pila de protocolos local.

Los MIH Commands remotos (Figura 2.9) son enviados por los usuarios MIH a la MIHF de la pila de protocolos asociada. Un comando MIH remoto entregado a la MIHF asociada es ejecutado por los niveles inferiores de la MIHF asociada como un comando de enlace; o es ejecutada por la propia MIHF asociada como un comando MIH.

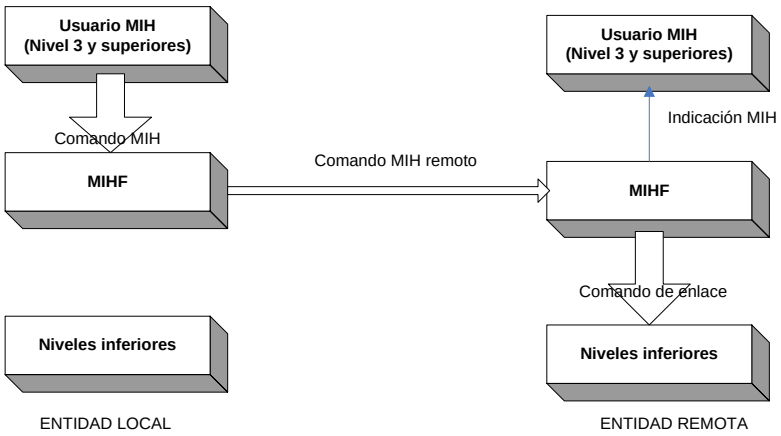

Figura 2.9: Comando MIH remoto

2.4.3.1 Comandos de Enlace

En la siguiente tabla (Tabla 2.4) se describen los Comandos de Enlace (Link Commands).

Comando	**Descripción**
Link_Capability_Discover	Petición y descubrimiento de la lista de link events y comandos de nivel de enlace soportados
Link_Event_Subscribe	Suscripción a uno o más eventos de un enlace
Link_Event_Unsubscribe	Baja de un conjunto de eventos de nivel de enlace
Link_Get_Parameters	Obtiene los parámetros medidos por el enlace activo: SNR, BER, RSSI
Link_Configure_Thresholds	Configura los umbrales para el reporte de eventos de enlace
Link_Action	Petición de una acción sobre la conexión a nivel de enlace

Tabla 2.4: Link Commands

2.4.3.2 Comandos MIH

En la siguiente tabla (Tabla 2.5) se describen los Comandos MIH (MIH Commands).

Evento	**(L)ocal (R)emoto**	**Descripción**
MIH_Link_Get_Parameters	L,R	Obtiene el estado de un enlace
MIH_Link_Configure_Thresholds	L,R	Configura los umbrales de los parámetros del enlace
MIH_Link_Actions	L,R	Controla el comportamiento de un conjunto de enlaces
MIH_Net_HO_Candidate_Query	R	El handover es iniciado por la red y manda una lista de posibles redes y sus PoAs asociados
MIH_MN_HO_Candidate_Query	R	Comando usado por el MN para pedir y obtener información sobre posibles redes candidatas para el handover
MIH_N2N_HO_Query_Resources	R	Este comando es enviado por la MIHF en servicio a la MIHF objetivo para la petición de recursos

MIH_MN_HO_Commit	R	Comando usado por el MN para notificar a la red en servicio la información de la red elegida
MIH_Net_HO_Commit	R	Comando usado por la red para notificar al MN la información de la red elegida
MIH_N2N_HO_Commit	R	Comando usado por la red en servicio para informar a la nueva red de que un MN va a trasladarse a ella, iniciar un contexto y preparar el handover
MIH_MN_HO_Complete	R	Notificación de la MIHF del MN a la MIHF de la nueva red o a la fuente indicando el estado de la finalización del handover
MIH_N2N_HO_Complete	R	Notificación de la MIHF de la nueva red o a la fuente a la otra MIHF indicando el estado de la finalización del handover

Tabla 2.5: MIH Commands

2.4.4 Media Independent Information Service

El Media Independent Information Service (MIIS) define una estructura mediante la que una MIHF, que reside en el MN o en la red, descubre y obtiene información de red dentro de un área geográfica que facilita la selección de red y los handovers.

El MIIS incluye soporte para varios Elementos de Información (Information Elements, IEs). Los IEs proporcionan información esencial para que la selección se haga de la mejor manera a la hora de decidir el handover.

El MIIS proporciona los medios para obtener información sobre niveles inferiores como mapas de redes vecinas u otros parámetros de enlace, al igual que de servicios de niveles superiores como la conectividad a Internet.

El Media Independent Information Service proporciona un mecanismo genérico que permite a un proveedor de servicios y a un usuario móvil intercambiar información de redes de acceso candidatas para el handover. Esta información incluye diferentes tecnologías como redes IEE802, redes 3GPP y redes 3GPP2. El MIIS también permite obtener esta información desde cualquier red.

La mayor parte de la información proporcionada por el Information Service es estática aunque los cambios en la configuración de la red también se tienen en cuenta. Los motivos que llevan a implementar este servicio son proporcionar los siguientes elementos:

- Información sobre las redes de acceso disponibles en una determinada área geográfica.

- Información de parámetros estáticos del nivel de enlace que ayudan a los nodos móviles en la selección de la red.

- Información sobre las capacidades de los diferentes PoAs reportados por las redes vecinas que ayuda a una configuración óptima del enlace radio para la conexión o la selección de redes.

- Indicaciones de los servicios que soportan las capas superiores de las diferentes redes de acceso y core, que ayudan a las decisiones sobre el handover.

2.4.4.1 Elementos de Información

Los IEs se clasifican en tres grupos:

1.- Información general e información específica de la red de acceso: Estos IEs dan una visión general de las diferentes redes que proporcionan cobertura en un área.

2.- Información específica de PoA: Proporcionan información sobre los diferentes PoAs para cada una de las redes de acceso disponibles.

3.- Otra información que es específica de la red de acceso, del servicio o del operador.

Cada IE ID es un valor de 32 bits. El espacio de nombres para los IEs se describe, a continuación, en la Tabla 2.6.

Rango	Descripción	Comentarios
0x00000000	Reservado	
0x00000001 – 0x1FFFFFFF	IEEE 802.21 IEs	Usado para los IEs definidos por IEEE 802.21
0x20000000 – 0x7FFFFFFF	IEs específicos del operador	Usado para IEs definidos por el operador
0x80000000 – 0x82FFFFFF	Reservado para desarrollo	Usado en desarrollo y pruebas
0x83000000 – 0xFFFFFFFF	Reservados	Uso futuro

Tabla 2.6: Espacio de nombres para los IEs

2.4.4.2 Representación de los elementos de información y métodos de consulta

El MIIS define dos métodos para representar IEs: representación binaria y representación RDF (Resource Description Framework). El MIIS también define dos métodos de consulta. Para consultas usando la representación binaria el método TLV (Type Length Value); para consultas usando la representación RDF el método SPARQL.

2.4.4.2.1 Representación binaria y consulta TLV

En el método de representación binario, los IEs se representan y codifican en forma TLV como muestra la siguiente figura (Figura 2.10).

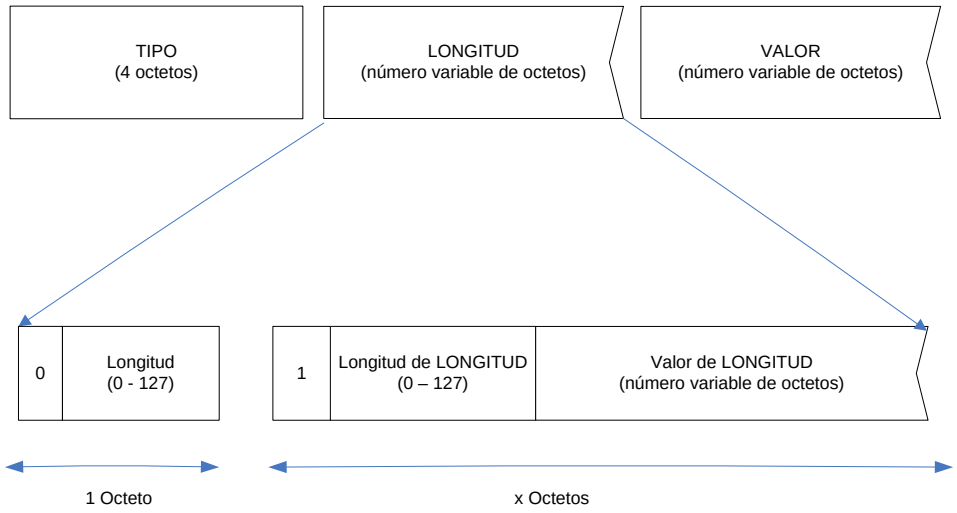

Figura 2.10: Representación TLV de IEs

Donde el campo de la longitud (Length) se codifica como:

- Caso 1: Si el número de octetos ocupados por el campo valor es menor que 128, el campo Longitud es siempre un octeto y el MSB del octeto se pone a 0. El resto de los bits indican la longitud de Valor.

- Caso 2: Si el número de octetos ocupados por el campo valor es 128, el campo Longitud es un octeto. El MSB se pone a 1 y el resto de bits a 0.

- Caso 3: Si el número de octetos ocupados por el campo valor es mayor que 128, el campo Longitud ocupa siempre mas de un octeto. El MSB del primer octeto se pone a 1 y los siete bits restantes indican el número de octetos que se usan para la longitud. El segundo y sucesivos octetos indican el tamaño del campo Valor.

2.4.4.2.2 Contenedores IE

En el método de representación binario se definen tres Contenedores IE:

- IE_CONTAINER_LIST_OF_NETWORKS: Contiene una lista de redes de acceso heterogéneas para una localización geográfica.

- IE_CONTAINER_NETWORK: contiene toda la información que describe una red de acceso.

- IE_CONTAINER_POA: Contiene toda la información que describe un PoA y opcionalmente uno o más IEs específicos de cada operador para PoAs.

2.4.4.2.3 Consultas TLV

Una consulta TLV incluye los siguientes parámetros opcionales para mejorar la consulta:

- QUERIER_LOC: Contiene las medidas de localización actuales del cliente, o si no son conocidas, direcciones de enlace que observa, que el IS (Information Server) será capaz de usar para estimar dichas medidas.

- NET_TYPE_INC: Indica el tipo de redes que el cliente quiere incluir en la respuesta.

- NETWK_INC: : Indica las redes de acceso específicas que el cliente quiere incluir en la respuesta.

- RPTL_TEMP: Da al IS una plantilla de la lista de IEs que se quieren incluir en la respuesta.

Cuando se recibe una petición binaria, el IS debe:

1.- Crear una lista de información de redes de acceso para el área dado.

2.- Si el parámetro RPTL_TEMPL no se incluye en la consulta, enviar la lista de información de redes de acceso en un IE_CONATINER_LIST_OF_NETWORKS en el mensaje MIH_Get_Information de respuesta.

3.- Si el parámetro RPTL_TEMPL se incluye en la consulta, extraer los IEs requeridos de la lista de información de redes de acceso usando las reglas definidas por RPT_TEMPL y enviarlos en el mensaje MIH_Get_Information de respuesta.

2.4.4.2.4 Representación RDF y consulta SPARQL

La representación RDF de los IEs se construye en formato XML. SPARQL se usa como método de consulta.

2.4.4.2.5 Esquema IS

Se usa un esquema en el IS de IEEE 802.21 para definir la estructura de cada elemento de información, como también para representar la relación entre ellos. El esquema IS IEEE 802.21 es soportado por cada MIHF que implementa el MIIS.

2.5 Service Access Points (SAPs) y Primitivas

La MIHF usa los siguientes SAPs para conectarse con otras entidades:

- SAPs dependientes del método de acceso:

  ● MIH_LINK_SAP: Interfaz abstracto dependiente del método de acceso de la MIHF con las capas inferiores de la pila de protocolos específica del modo de acceso.

  ● MIH_NET_SAP: Interfaz abstracto dependiente del método de acceso de la MIHF que proporciona servicios de transporte sobre el plano de datos en el nodo local, dando soporte al intercambio de información y mensajes MIH con la MIHF remota.

- SAPs independientes del método de acceso:

  ● MIH_SAP: Este SAP define el interfaz independiente del método de acceso entre la MIHF y los usuarios MIH.

Los SAPs son definidos como un conjunto de primitivas. Las primitivas definen los servicios. Cada parámetro se define usando tipos de datos abstractos. Estos tipos indican el valor semántico de ese parámetro. Los parámetros definidos dentro de la subclaúsula para una primitiva en particular son definidos y usados por esa primitiva. Varios de los tipos de datos abstractos se usan en definiciones de múltiples primitivas.

2.5.1 SAPs dependientes del método de acceso

2.5.1.1 MIH_LINK_SAP

Las primitivas definidas como parte del MIH_LINK_SAP se describen en la Tabla 2.7.

Primitivas	**Categoría de Servicio**	**Descripción**
Link_Detected	Evento	Se ha detectado un nuevo enlace
Link_Up	Evento	La conexión a nivel 2 ha sido establecida
Link_Down	Evento	La conexión a nivel 2 se ha perdido
Link_Parameters_Report	Evento	Los parámetros de enlace han cruzado el umbral especificado
Link_Going_Down	Evento	La pérdida de la conexión a nivel 2 es inminente
Link_Handover_Inminent	Evento	El Handover a nivel 2 es inminente
Link_Handover_Complete	Evento	El Handover a nivel 2 ha sido completado
Link_PDU_Transmit_Status	Evento	Estado de la transmisión de una PDU
Link_Capability_Discover	Comando	Consulta y descubrimiento de la lista de eventos y comandos de enlace soportados
Link_Event_Subscribe	Comando	Suscripción a la notificación de eventos
Link_Event_Unsubscribe	Comando	Baja de la notificación de eventos
Link_Get_Parameters	Comando	Petición de parámetros del medio
Link_Configure_Thresholds	Comando	Configurar los umbrales del enlace para los eventos
Link_Action	Comando	Petición de una acción en una conexión a nivel de enlace

Tabla 2.7: Primitivas MIH_LINK_SAP

2.5.1.2 MIH_NET_SAP

La primitiva definida para el MIH_NET_SAP se describe en la Tabla 2.8

Primitivas	**Categoría de Servicio**	**Descripción**
MIH_TP_Data	Comunicación de Red	Transferencia de datos

Tabla 2.8: Primitiva MIH_NET_SAP

2.5.2 SAPs independientes del método de acceso: Primitivas MIH_SAP

Las primitivas definidas como parte del MIH_SAP se describen en la Tabla 2.9

Primitivas	Categoría de Servicio	Descripción
MIH_Capability_Discover	Gestión de Servicio	Descubre la lista de eventos y comandos soportados por la MIHF
MIH_Register	Gestión de Servicio	Registro con una MIHF remota
MIH_DeRegister	Gestión de Servicio	Desregistro de una MIHF remota
MIH_Event_Subscribe	Gestión de Servicio	Suscripción a la notificación de eventos MIH
MIH_Event_Unsubscribe	Gestión de Servicio	Baja de la notificación de eventos MIH
MIH_Link_Detected	Evento	Se ha detectado un nuevo enlace
MIH_Link_Up	Evento	La conexión a nivel 2 ha sido establecida
MIH_Link_Down	Evento	La conexión a nivel 2 se ha perdido
MIH_Link_Parameters_Report	Evento	Los parámetros del enlace han cruzado el umbral predefinido
MIH_Link_Going_Down	Evento	La conexión a nivel 2 se va a caer
MIH_Link_Handover_Inminent	Evento	El handover de nivel 2 es inminente
MIH_Link_Handover_Complete	Evento	El handover de nivel 2 ha sido completado
MIH_Link_PDU_Transmit_Status	Evento	Estado de transmisión de una PDU
MIH_Link_Get_Parameters	Comando	Obtiene el estado de un enlace
MIH_Link_Configure_Thresholds	Comando	Configura los umbrales de los parámetros del enlace
MIH_Link_Actions	Comando	Controla el comportamiento de un conjunto de enlaces
MIH_Net_HO_Candidate_Query	Comando	Inicia el handover
MIH_MN_HO_Candidate_Query	Comando	Inicia la consulta del MN de las redes candidatas
MIH_N2N_HO_Query_Resources	Comando	Consulta los recursos disponibles en la red
MIH_MN_HO_Commit	Comando	Notificar a la red en servicio la información de la red elegida
MIH_Net_HO_Commit	Comando	La red ha asignado el handover
MIH_N2N_HO_Commit	Comando	Notificar a la red elegida que la red en servicio ha asignado el handover
MIH_MN_HO_Complete	Comando	Inicia la notificación de la finalización del handover por parte del MN
MIH_N2N_HO_Complete	Comando	El handover ha sido realizado
MIH_Get_Information	Información	Petición para obtener información de un repositorio
MIH_Push_Information	Información	Notificar al MN las políticas de operación y otra información

Tabla 2.9: Primitivas MIH_SAP

Las primitivas de comando definidas en MIH_SAP indican su destino ya sea la MIHF local o una MIHF remota. Para el caso remoto, la MIHF local procesará primero la primitiva para crear un mensaje MIH y luego mandará el mensaje a la MIHF destino asociada para su ejecución.

2.6 Protocolo MIH

Las MIHFs en los MNs y en las entidades de red se comunican usando los mensajes del protocolo MIH. El protocolo MIH define el formato de los mensajes que se intercambian las MIHFs asociadas y de los mecanismos que dan soporte a la entrega de dichos mensajes. Estos mensajes se basan en las primitivas que son parte de los servicios MIH.

El protocolo Media Independent Handover define el intercambio de mensajes entre dos entidades MIHF que da soporte a los servicios MIH. Una transacción MIH se identifica por una secuencia de mensajes con el mismo Transaction-ID transmitido/recibido hacia/desde un MIHD ID remoto.

En cualquier momento dado, un nodo MIH no puede tener más de una transacción pendiente en cada dirección con un cierto nodo MIH asociado, es decir, que el nodo MIH debe esperar a que cualquier transacción pendiente termine antes de comenzar una nueva transacción con el mismo nodo asociado.

Los mensajes son enviados sobre el plano de datos usando un mecanismo de transporte apropiado tanto de nivel 2 como de nivel 3. Para el transporte a nivel 3 se usan los protocolos TCP, UDP y SCTP sobre IP. Para el transporte a nivel 2 las PDUs deben identificarse usando el campo EtherType para el protocolo MIH (valor 8917).

El protocolo Media Independent Handover debe implementarse de forma adecuada de acuerdo al estándar en los nodos que dan soporte al servicio MIH ya que es el encargado de la comunicación entre las entidades involucradas en el handover y éstas pertenecen a redes de distinta naturaleza (y fabricantes) que deben intercambiar mensajes de señalización comunes que han de ser debidamente procesados para el correcto funcionamiento del servicio.

2.6.1 Servicio de asentimiento (ACK)

Este servicio se debe usar cuando el transporte usado para la comunicación no proporciona un servicio fiable.

La MIHF fuente solicita el asentimiento del mensaje para asegurarse de que el destinatario ha recibido el mensaje MIH.

El servicio de asentimiento MIH utiliza dos bits de información que se definen para el uso exclusivo de este servicio y que se sitúan en la cabecera del mensaje MIH. El bit ACK-Req es fijado por el nodo MIH fuente y el bit ACK-Rsp es fijado por el nodo MIH destino.

El nodo fuente tiene que iniciar el timer de retransmisión después de enviar un mensaje MIH cuando el bit ACK-Req está activado y guardar una copia del mensaje mientras el timer esté activo. Si no se recibe el ACK antes de que expire el timer, el nodo fuente debe retransmitir el mensaje guardado con el mismo Message-ID y Transaction-ID (con el bit ACK-Req activado). Si el nodo fuente recibe el ACK antes

de que expire el timer (del primer mensaje o de cualquier retransmisión) entiende que el mensaje ha sido recibido, resetea el timer y borra la copia del mensaje. Si durante la retransmisión recibe un ACK de cualquier intento de transmisión previo, entiende que el mensaje se ha recibido correctamente y no tiene que esperar más ACKs para dicho mensaje. El mensaje se retransmite hasta que se recibe un ACK o se alcanza el número máximo de retransmisiones fijado.

Cuando el nodo MIH destino recibe un mensaje MIH con el bit ACK-Req activado, entonces dicho nodo devuelve un mensaje MIH con el bit ACK-Rsp activado copiando el Message-ID y el Transaction-ID del mensaje recibido. El mensaje con el bit ACK-Rsp activado sólo consta de la cabecera MIH y no tiene payload. Si el destino procesa el mensaje inmediatamente y está disponible la respuesta, el bit ACK-Req es activado en el mensaje de respuesta correspondiente.

El nodo destino responde con un ACK para mensajes MIH duplicados (mensajes con el mismo Transaction-ID) que tienen el bit ACK-Req activado. Sin embargo, los mensajes duplicados no se procesan si ya han sido procesados anteriormente. Si un nodo MIH destino recibe un mensaje con el bit ACK-Req desactivado no se realiza ninguna acción relacionada con el servicio de asentimiento.

En todos los casos, el mensaje MIH de una transacción se procesa una sola vez en el destino independientemente del número de mensajes recibidos con el bit ACK-Req activado.

2.6.2 Máquinas de Estado

Un nodo que tiene un mensaje nuevo disponible para ser enviado relativo una nueva transacción se denomina fuente de la transacción e inicia la transaction source state machine. Un nodo que recibe un mensaje relativo a un a nueva transacción se denomina nodo destino de la transacción e inicia la destination transaction state machine.

Si se está usando el servicio de asentimiento por parte de la fuente y/o el destino de una transacción, la ACK.Requestor y/o la ACK-Responder state machine es iniciada. La máquina de estado relacionada con el ACK corre en paralelo con las transaction source/destination state machines.

Todas las instancias de las máquinas de estado relativas a una transacción tienen acceso a las variables, constantes y procedimientos inter-state-machine, que no son accesibles por las máquinas de estado relativas a otras transacciones. Las variables inter-state-machine permiten la comunicación entre máquinas de estado para una transacción dada. No se puede dar el caso de que dos o más máquinas de estado para una transacción dada escriban la misma variable inter-state-machine al mismo tiempo. A las variables, constantes y procedimientos intra-state-machine sólo puede acceder una máquina de estado para una transacción dada.

2.6.2.1 Variables, constantes y procedimientos

2.6.2.1.1 Variables inter-state-machine

Estas variables están disponibles para más de una máquina de estado relativa a una instancia de una transacción y se usan para realizar la comunicación inter-state-machine y la inicialización de funciones dentro de la transacción.

Las variables Exported son variables de tipo inter-state-machine que también pueden ser leídas y escritas desde entidades externas a las máquinas de estado.

Las variables inter-sate-machine se definen a continuación. El valor que aparece entre paréntesis es el tipo de dato de la variable y que se define en el estándar:

- Opcode (OPCODE): Código de operación
- MID (MID): Identificador del mensaje
- AckRequestorStatus (ENUMERATED): Indica el estado de la ACK requestor state machine. Esta variable es inicializada por la transaction source state machine y modificada por la ACK requestor state machine. Sus posibles valores son:

  - 1 ONGOING
  - 2 SUCCESS
  - 3 FAILURE

- TransactionStopWhen (UNSIGNED_INT(1)): Timer para detener la transacción.
- RetransmissionWhen (UNSIGNED_INT(1)): Timer para la retransmisión de un mensaje.

Las variables Exported se definen a continuación:

- TID (TID): Identificador de transacción.
- MyMihfID (MIHF_ID): El MIHF ID del nodo MIH
- PeerMihfID (MIHF_ID): El MIHF ID del nodo MIH asociado
- MsgIn (MIH_MESSAGE): Un mensaje recibido válido.
- MsgInAvail (BOOLEAN): Esta variable es puesta a TRUE por una entidad externa las máquinas de estado cuando un mensaje recibido es válido y está disponible para una transacción. Se inicializa a FALSE por una entidad externa.
- MsgOut (MIH_MESSAGE): Un mensaje válido, generado por la MIHF local, para ser enviado a una MIHF remota.
- MsgOutAvail (BOOLEAN): Esta variable es puesta a TRUE por una entidad externa las máquinas de estado o por la transaction source o destination state machine cuando un mensaje está disponible par ser enviado en una transacción. Se inicializa a FALSE por una entidad externa.
- TransactionStatus (ENUMERATED): Indica el estado e la transacción. Es escrita por la maquina de estado y leída por la MIHF. Sus posibles valores son:

  - 1 ONGOING
  - 2 SUCCESS
  - 3 FAILURE

- StartAckRequestor (BOOLEAN): Se inicializa a FALSE por una entidad externa. Cuando se pone a TRUE (por su transaction source o destination state machine) se inicia una instancia de la ACK requestor state machine.

- StartAckResponder (BOOLEAN): Se inicializa a FALSE por una entidad externa. Cuando se pone a TRUE (por su transaction source o destination state machine) se inicia una instancia de la ACK requestor state machine.

2.6.2.1.2 Variables intra-state-machine

- DUP (MIH_MESSAGE): Contiene un mensaje MIH ya enviado. Es utilizada por la ACK responder state machine.

- ACK (MIH_MESSAGE): Contiene un mensaje MIH con el bit ACK-Rsp activado y el mismo MID y Transaction-ID del mensaje que asiente. Es utilizada por la ACK responder state machine.

- RtxCtr (UNSIGNED_INT(1)): Número de retransmisiones de un determinado mensaje. Es utilizada por la ACK requestor state state machine.

2.6.2.1.3 Constantes inter-state-machine

- TransactionLifeTime: Tiempo máximo entre el inicio y la finalización de una transacción

- Request: Un OPCODE de valor 0x01

- Response: Un OPCODE de valor 0x02

- Indication: Un OPCODE de valor 0x03

2.6.2.1.4 Constantes intra-state-machine

- RetransmisionInterval: Intervalo de tiempo entre dos transmisiones consecutivas de un mismo mensaje.

- MaxRtxCtr: Máximo número de veces que un mensaje será retransmitido, si se dan las condiciones de retransmisión.

2.6.2.1.5 Procedimientos inter-state-machine

- BOOLEAN Process (MIH_MESSAGE): Procesa el mensaje recibido y que es pasado como variable de entrada. Devuelve TRUE si hay un mensaje de respuesta listo para ser enviado. En cualquier otro caso el valor devuelto es FALSE.

- void Transmit (MIH_MESSAGE): Transmite el mensaje que le es pasado como variable de entrada.

- MIHF_ID SrcMIHF_ID (MIH_MESSAGE): Extrae el campo Source Identifier del mensaje pasado como variable y devuelve el valor de dicho campo.

- MIHF_ID DstMIHF_ID (MIH_MESSAGE): Extrae el campo Destination Identifier del mensaje pasado como variable y devuelve el valor de dicho campo.

- void setMIHF_ID (MIH_MESSAGE, MIHF_ID, MIHF_ID): Inserta el Source Identifier y el Destination Identifier en el mensaje MIH. El primer MIHF_ID se corresponde a la fuente y el segundo al destino.

2.6.2.2 Transaction Source State Machine

La transaction source state machine se inicia cuando un mensaje relativo a una nueva transacción está listo para ser enviado (MsgOutAvail está a TRUE). La transacción termina cuando hay una transición al estado SUCCESS y las máquinas de estado relativas al servicio de asentimiento han terminado (si fueron iniciadas); o si hay una transición al estado FAILURE. Una instancia de la transaction source state machine deja de existir una vez que la variable TransactionStatus toma el valor SUCCESS o FAILURE.

2.6.2.3 Transaction Destination State Machine

La transaction destination state machine se inicia cuando un mensaje relativo a una nueva transacción es recibido (MsgInAvail está a TRUE).

La transacción termina cuando se transita al estado FAILURE o al estado SUCCESS y las máquinas de estado relativas al servicio de asentimiento han terminado (si fueron iniciadas). Una instancia de la transaction destination state machine deja de existir una vez que la variable TransactionStatus toma el valor SUCCESS o FAILURE.

2.6.2.4 ACK Requestor State Machine

La ACK requestor state machine es iniciada cuando la variable StartAckRequestor cambia su valor a TRUE en la source/destination state machine. Termina cuando transita al estado FAILURE o al estado SUCCESS. Una instancia de la ACK requestor state machine deja de existir una vez que la variable AckRequestorStatus toma el valor SUCCESS o FAILURE o su transaction source/destination state machine ha dejado de existir.

2.6.2.5 ACK Responder State Machine

La ACK responder state machine es iniciada cuando la variable StartAckResponder cambia su valor a TRUE en la source/destination state machine. Termina cuando transita al estado FAILURE o al estado SUCCESS. Una instancia de la ACK responder state machine deja de existir una vez que la variable su transaction source/destination state machine ha dejado de existir.

2.6.3 Identificadores del protocolo MIH

Se usan dos identificadores:

- MIHF Identifier (MIHF ID): Identifica de manera única a una entidad MIHF. Es usado en todos los mensajes del protocolo MIH. Es asignado durante la configuración. Es de tipo MIHF_ID.

- Transaction Identifier (Transaction_ID): es usado para identificar la correspondencia entre la petición y la respuesta. También es necesario para identificar la correspondencia entre la petición, la respuesta o la indicación y su correspondiente ACK. Es creado por el nodo que inicia la transacción y se introduce en la cabecera del

mensaje MIH. Se define como un entero sin signo de 16 bits cuyo valor es único entre todas las transacciones pendientes para un transmisor y un receptor dado.

2.6.4 Formato de trama

En los mensajes del protocolo MIH, todas las definiciones TLV son alineadas a octeto y no es necesario hacer padding. El payload de los mensajes del protocolo MIH lleva el identificador de la fuente (Source MIHF Identifier TLV) y el identificador del destino (Destination MIHF Identifier) seguidos de campos TLV que dependen del servicio que se esté utilizando.

Figura 2.11: Formato de Trama del protocolo MIH

La cabecera del protocolo (Figura 2.12) lleva información imprescindible que está presente en todas las tramas y que es usada para analizar la trama MIH.

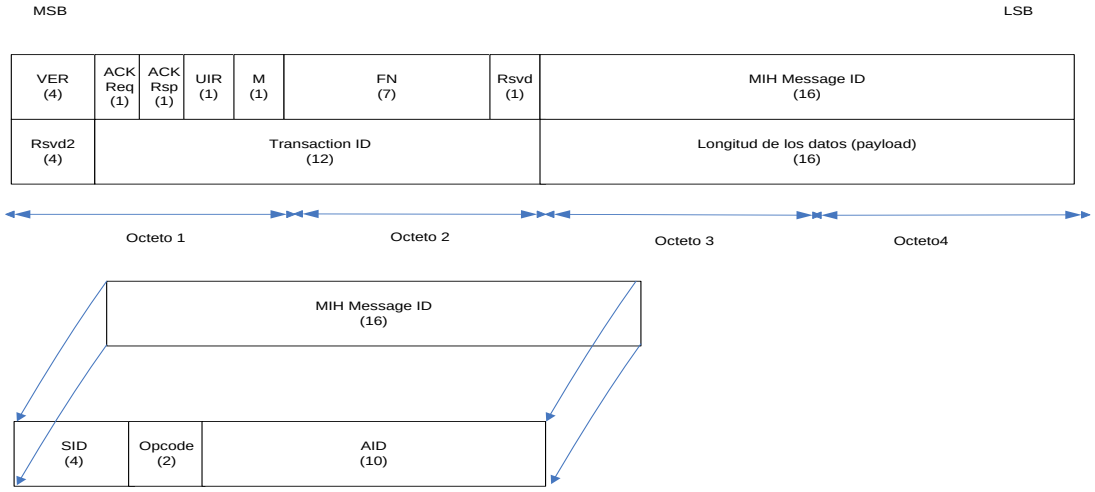

Figura 2.12: Formato de cabecera de la trama MIH.

A continuación se describen los campos de la cabecera:

- Version (4 bits): Versión del protocolo MIH.
- ACK-Req (1 bit): Se usa para solicitar el asentimiento del mensaje.
- ACK-Rsp (1 bit): Usado para responder a una petición de asentimiento del mensaje.
- Unauthenticated information request, UIR (1 bit): Usado por el MIH Information Service para indicar si el mensaje es enviado en el estado pre-autenticado/pre-asociación ya que la longitud de la respuesta puede verse limitada.

- More fragment, M (1 bit): Indica que el mensaje es un fragmento al que seguirá otro fragmento. Se pone a 0 si el mensaje no es fragmentado o es el último fragmento, y se diferencia cada uno de los casos por medio del campo FN. Se pone a 1 para fragmentos que no son los últimos.

- Fragment number, FN (7 bits): Representa el número de secuencia de un fragmento. Empieza por 0. El máximo FN es 127. Se pone a 0 cuando el mensaje no es fragmentado.

- Reserved1 (1 bit): Campo reservado. Cuando no se usa se pone a 0.

- MIH message ID, MID (16 bits): Consta de tres campos:

  ● Service identifier, SID (4 bits): Identifica los diferentes servicios MIH. Sus posibles valores son:

    1: Service Management
    2. Event Service
    3. Command Service
    4. Information Service

  ● Operation code, Opcode (2 bits): Tipo de operación a realizar con respecto al SID. Sus posibles valores son:

    1: Request
    2. Response
    3. Indication

  ● Action identifier, AID (10 bits): Indica la acción a realizar teniendo en cuenta el SID.

- Reserved2 (4 bits): Campo reservado. Cuando no se usa se pone a 0.

- Transaction ID (12 bits): Usado para saber la correspondencia entre petición y respuesta, y también entre una petición, una respuesta o una indicación con un ACK.

- Variable payload length (16 bits): Indica la longitud del payload de la trama MIH. No se incluye la longitud de la cabecera MIH.

2.6.5 Codificación TLV de parámetros del mensaje

La codificación TLV usada para los parámetros de los mensajes del protocolo MIH (Figura 2.13) consta de los siguientes campos:

- Tipo, Type (1 octeto): Tipo del parámetro. Debe ser único dentro del protocolo MIH.

- Longitud, Length (variable): Longitud del campo Valor (Value) del parámetro. La longitud sigue la misma codificación que las reglas descritas en el apartado 2.4.3.2.1.

- Valor, Value (variable): Valor del parámetro.

Tipo (1 octeto) Tipo del parámetro	Longitud (variable) Longitud del campo Valor	Valor (variable) Valor del parámetro

Figura 2.13: Codificación TLV de los parámetros de un mensaje

2.6.6 Mensajes MIH

Todos los mensajes MIH llevan el campo source MIHF ID seguido del campo destination MIHF ID en los dos primeros TLVs del payload del mensaje.

Los campos opcionales pueden o no ser enviados y el receptor los tratará adecuadamente (usándolos si los considera necesarios) si están presentes.

Cuando se recibe un mensaje de petición la MIHF debe enviar su correspondiente mensaje de respuesta.

Cuando el mensaje recibido tenga una cabecera MIH no válida, o no contenga el source/destination MIHF ID o el MID no sea válido, será descartado sin indicar nada al nodo fuente. Si se reciben TLVs no válidos, éstos serán ignorados.

3. Protocolo MIH: Implementación en C

3.1 Introducción

La implementación de las funciones y programas que definen el prototipo IEEE 802.21 desarrollado en este proyecto se ha realizado mediante el lenguaje de programación C [8].

En este apartado se realiza una descripción detallada de cada una de las primitivas definidas, así como de la estructuras de datos y programas desarrollados, para la implementación del handover iniciado por el MN o por la red.

La implementación del prototipo consta de un archivo de cabecera (.h), que contiene definiciones, y cuatro archivos de código C (.c) que contienen los programas principales:

- MIH_Protocol.h: Este archivo contiene:

  ● Definiciones de las estructuras de datos, variables y funciones globales necesarias par el funcionamiento del prototipo.
  ● Definición de las primitivas necesarias para la realización del handover iniciado por el MN y por la red
  ● Definición de las máquinas de estado que se encargan de las transacciones.

- Programas principales: Cada uno de los programas principales se ejecuta en el nodo correspondiente y es el encargado de llevar el control en el nodo de las distintas fases del handover. Estos programas principales son:

  ● Mobile_Node.c: Nodo que se mueve entre las distintas redes
  ● Serving_Network.c: Red que da servicio al MN
  ● Candidate1.c: Primera red candidata para el handover
  ● Candidate2.c: Segunda red candidata para el handover

3.2 Estructuras, variables y funciones globales

3.2.1 Estructuras de datos

Durante el handover es necesario guardar cierta información para su posterior uso, para ello se han definido estructuras de datos de tipos struct, con campos de distintos tipos según la información a guardar. Estas estructuras son:

- Header_M: Estructura donde se guardan los datos de la cabecera de los mensajes MIH. Se definen 13 campos dentro de la estructura para guardar cada uno de los valores descritos en el apartado 2.6.4

- Candidate_Query_Response_Information: Guarda la información recibida por el MN en el mensaje MIH_MN_HO_Candidate_Query Response. Se definen tres campos dentro de la estructura para guardar cada uno de los valores de los distintos campos TLV del mensaje MIH recibido.

- Candidate_Query_Request_Information: Guarda la información recibida por el la Serving Network en el mensaje MIH_Net_HO_Candidate_Query Response o bien en el mensaje MIH_MN_HO_Candidate_Query_Request Message. Se definen siete campos donde se guardan diferentes valores de los campos TLV del mensaje MIH recibido.

- Commit_Information: Guarda la información recibida por el MN en el mensaje MIH_MN_HO_Candidate_Query Response. Se definen dos campos para guardar datos usados en la fase de preparación de recursos del handover.

Para representar valores de estados se definen varios tipos de datos enumerados, cuyos valores se corresponden con los definidos en el estándar. Estos tipos de datos son Bit_Activate, Status, HO_Result, HO_Status, Link_AC_Result y Transaction_Status.

Además también se define un tipo enumerado (BOOLEAN) para representar valores booleanos.

3.2.2 Funciones auxiliares

Existen varios tipos de funciones auxiliares según los servicios que proporcionan. Estos tipos son:

- Funciones para el cálculo de longitudes: Cada una de estas funciones calcula la longitud de la estructura que se le pasa como parámetro. El parámetro de entrada es un puntero que apunta al comienzo de la cadena de bytes de la cual queremos calcular la longitud y que consta de la codificación en bytes del tipo de dato, según el estándar, del cual queremos calcular su longitud. Para ello se obtienen los campos, a través de la aritmética de punteros, que contienen longitudes de los diversos elementos del tipo de dato, y se van sumando dichos campos para obtener la longitud total. En el caso de las listas, se repite la operación anterior tantas veces como número de elementos contenga la lista, mediante el uso del bucle for.

- Funciones de muestra de datos: Cada una de estas funciones imprime en pantalla los distintos campos de los que consta la estructura de datos que se le pasa como parámetro y que se corresponde con una cadena de bytes que representa la codificación de un determinado tipo de dato definido en el estándar. Mediante el uso de estas funciones, que son utilizadas por las primitivas, se obtiene la información de forma detallada de cada uno de los campos de los mensajes recibidos por el nodo, así como de información de las distintas acciones que se están llevando a cabo durante el handover. Para obtener los valores de los distintos campos se va recorriendo cada uno de esos campos mediante el uso de punteros auxiliares hasta el final de la estructura. Los valores de estos campos se presentan bien de forma numérica, o de forma descriptiva si el valor numérico se corresponde a un determinado estado.

- Funciones para la obtención de la fuente y el destino de un mensaje MIH: Estas funciones se denominan SrcMIHFID y DstMIHFID, y obtienen el MIHF_ID de la fuente y el destino, respectivamente, a partir del mensaje MIH que se le pasa como parámetro. Devuelven el puntero a una estructura donde se guardan los datos. Para ello, se utiliza un puntero que recorre la cadena de bytes y que obtiene la longitud del campo a partir del byte o bytes correspondientes dentro de la estructura TLV que lleva la información del identificador y que es definida en el estándar. Se corresponden con los procedimientos descritos en el apartado 2.6.2.1.5.

- Funciones para el procesado de mensajes MIH: Estas dos funciones se denominan process_message () y process_message_dest (), y son utilizadas por la transaction source y destination state machines respectivamente, para el procesado de los mensajes recibidos por un nodo. Su estructura de implementación en C es la misma y consiste en recuperar, a partir de la cabecera del mensaje, los campos SID, Opcode y AID, mediante los cuales se obtiene el tipo de mensaje recibido. La información de la cabecera del mensaje MIH se recupera de la estructura Header_M, que previamente ha sido introducida cuando el mensaje es recibido. Mediante la aritmética de punteros se obtienen los distintos campos en función del tipo de mensaje y se guardan en las variables definidas para el almacenamiento de datos. Posteriormente se realiza la llamada a la primitiva que corresponda en función del tipo de mensaje recibido, utilizando para ello los datos obtenidos en el mensaje. Se corresponden con el procedimiento descrito en el apartado 2.6.2.1.5

- Funciones de liberación de memoria: Estas funciones son utilizadas por la máquina de estados para liberar la memoria reservada durante una transacción. Libera la memoria que ha sido reservada para guardar los mensajes MIH y los identificadores de la fuente y el destino una vez se ha acabado la transacción en curso. Estas funciones se denominan free_memory_source (utilizada por la Transaction source state machine) y free_memory_destination (utilizada por la Transaction destination state machine).

3.2.3 Variables globales

Estas variables son definidas en el archivo de cabecera MIH_Protocol.h y pueden ser utilizadas por todas las primitivas, así como, por los distintos programas principales para cada uno de los nodos. Según su función se dividen en los siguientes grupos:

- Variables para la gestión de los sockets: Hay tres tipos de variables para la gestión de los sockets [9] [10]:

● Descriptores de los sockets: Son de tipo entero (int), y almacenan el identificador del socket que se obtiene mediante la llamada a la función socket().

● Estructuras de información de los sockets: Son de tipo sockaddr_in6, y almacenan la información sobre la conexión. Cada una de estas variables se inicializan en el programa principal que se ejecuta en cada nodo, especificando la dirección IPv6 y el puerto que corresponde a cada conexión.

● Longitud de las estructuras de información: Son de tipo entero (int) y guardan el tamaño en bytes de la cada una de las estructuras de información de los sockets indicadas anteriormente.

- Variables para la gestión de los hilos de ejecución: Son de tipo pthread_t, se denominan idSourceTX e idDestinationRX, y almacenan el identificador del hilo que ejecuta la instancia de la Transaction source state machine y la Transaction source state machine. Además se define una variable de tipo entero (int) para la gestión de los errores de los hilos (error).

- Variables para el almacenamiento de mensajes e información de fuente y destino: Para el almacenamiento de mensajes MIH y de los identificadores de la fuente y el destino se utilizan punteros de tipo carácter (unsigned char) de tal forma que se asigna la memoria necesaria de forma dinámica y se libera una vez acabada la transacción en curso. Además se utilizan variables de tipo entero (int) para guardar longitudes de mensajes MIH, y estructuras de tipo Header_M (apartado 3.2.1) donde se guardan los datos de la cabecera del mensaje MIH.

- Variables para el control del programa: Representan las variables inter-state-machine descritas en el apartado 2.6.2.1.1 y son de tipo boolean. Además se definen variables, también booleanas, para controlar el paso por las distintas fases del handover (Resource_Availability_Check, Resource_Preparation y Resource_Release) y la correcta realización de cada fase y si es necesario parar el handover por un fallo (stop_handover). También se define una variable (source_tx_required) que controla si es necesario abrir una instancia de la transaction source state machine en los casos donde el mensaje de respuesta no está disponible inmediatamente y es necesario consultar a otros nodos para construirlo. Dentro de este grupo es definida la variable (number_of_candidate_networks), de tipo entero (int), que da información de sobre la red candidata que ha mandado el mensaje para que la primitiva que la usa ejecute un código determinado en función de su valor.

- Variables para la gestión de las máquinas de estados: Representan las variables inter-state-machine descritas en el apartado 2.6.2.1.1 que están relacionadas con las máquinas de estados. Su tipo varía entre unsigned char, unsigned short y unsigned int dependiendo de los bytes necesarios para su almacenamiento según los datos definidos en el estándar.

- Variables para el almacenamiento de información del handover: Son variables de distintos tipos según la información que es necesario almacenar. Esta información se almacena durante el handover para su utilización durante las fases del mismo.

- Otras variables globales auxiliares: son de tipo booleano y se utilizan para saber si la memoria reservada para los mensajes ha sido ya liberada o no.

3.2.4 Máquinas de estado

Las máquinas de estado (source y destination state machine) son utilizadas para controlar las transacciones necesarias durante el handover. Cada vez que un nodo dispone de un mensaje MIH, relativo a una nueva transacción, inicia una instancia de la source transaction state machine, la cual cesa cuando se recibe el mensaje de respuesta o ha habido algún error durante la transacción. Cada vez que un nodo recibe un mensaje MIH relacionado a una nueva transacción se inicia una instancia de la destination source state machine, esta instancia cesará cuando se reciba el mensaje de asentimiento del mensaje de respuesta que previamente se ha enviado, y que está relacionado con una petición previa.

El protocolo de transporte utilizado en este proyecto es UDP, por lo tanto, al ofrecer un servicio no fiable, es necesario implementar el servicio de asentimiento descrito en el estándar. De esta forma, cada uno de los mensajes enviados por los nodos ha de ser asentido para que la transacción sea correcta.

3.2.4.1 Transaction source state machine

El diagrama de flujo de la implementación en C de la transaction source state machine es el representado en la siguiente figura (Figura 3.1):

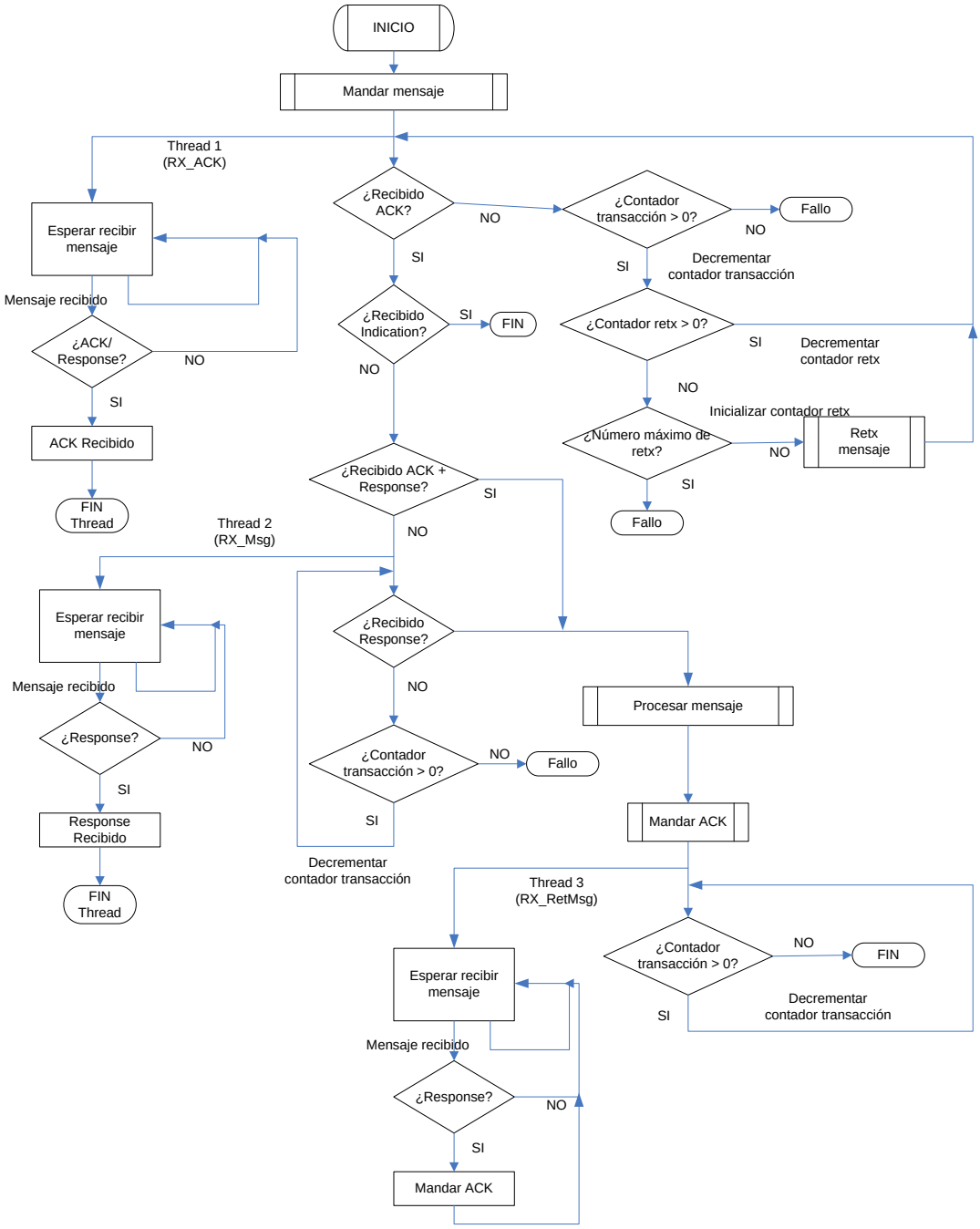

Figura 3.1: Diagrama de flujo C de la Transaction Source State Machine

Cuando está disponible un nuevo mensaje correspondiente a una nueva transacción se lanza un thread correspondiente a la transaction source state machine.

La transaction source state machine comienza enviando el mensaje mediante la función send de C para, a continuación, lanzar un thread (thread 1, RX_ACK) que se encarga de esperar el ACK del mensaje enviado, mediante un bucle y utilizando para ello la función C recvfrom, informando al thread principal de la recepción del ACK. Mientras, el thread principal se encarga de comprobar mediante un bucle si se ha recibido dicho ACK, decrementando el contador de transacción y comprobando si ha expirado el timer de dicha transacción, además de comprobar si es necesario

retransmitir el mensaje o se ha llegado al número máximo de retransmisiones. Si expira el tiempo sin recibir el ACK se da por fallida la transacción y se informa al usuario de la transaction source state machine mediante el cambio de valor de la variable TransactionStatus.

Si se recibe el mensaje de respuesta con el bit ACK_rsp a 1, se pasa a procesar el mensaje ya que se incluye el asentimiento del mensaje enviado.

Si se recibe el ACK y no se ha recibido la respuesta, se lanza un thread (thread 2, RX_Msg) que se encarga de esperar el mensaje de respuesta a la petición mediante la función C recvfrom, informando al thread principal de la recepción de la respuesta. El thread principal comprueba si se ha recibido la respuesta, decrementa el contador que indica el tiempo de transacción y comprueba que dicho tiempo no se ha agotado. Si esto sucede se da la transacción por finalizada y se indica que ha habido un error mediante el valor de la variable TransactionStatus.

Cuando se recibe la respuesta se pasa a procesar el mensaje recibido mediante la función process_message, para posteriormente mandar el ACK de la respuesta recibida. Por último, se lanza un thread (thread 3, RX_RetMsg) que se encarga de volver a retransmitir el ACK si se recibe una respuesta que ya ha sido procesada (y que no se vuelve a procesar). El thread principal se encarga de decrementar el contador de transacción hasta el fin de la misma, indicando que la transacción ha acabado correctamente.

3.2.4.2 Transaction destination state machine

El diagrama de flujo de la implementación en C de la transaction destination state machine es el representado en la siguiente figura (Figura 3.2):

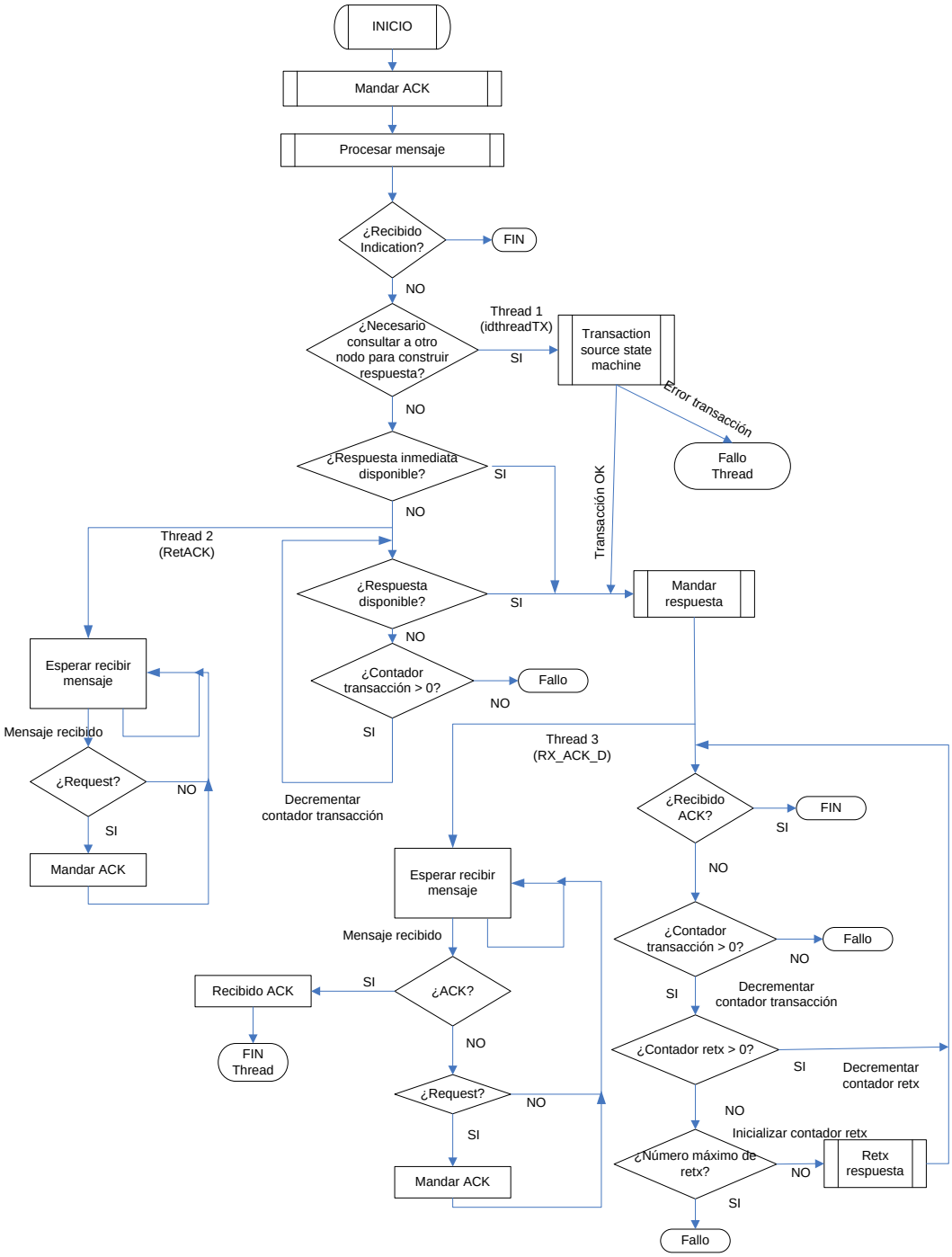

Figura 3.2: Diagrama de flujo C de la Transaction Destination State Machine

Cuando se recibe un nuevo mensaje correspondiente a una nueva transacción se lanza un thread correspondiente a la transaction destination state machine.

Lo primero que se hace es enviar el ACK del mensaje recibido para informar al nodo fuente que se ha recibido correctamente dicho mensaje para posteriormente pasar a procesar el mensaje mediante la función process_message_dest.

Se comprueba si es necesario intercambiar información con otros nodos antes de construir la respuesta. Si es así, se lanza una instancia de la transacción source state machine (thread 1, threadTX) para obtener la información necesaria para construir la respuesta a la petición. Si dicha transacción acaba correctamente se procede a mandar el mensaje de respuesta al nodo fuente de la petición. Si por el contrario la transacción falla, se informará de que ha habido un error y la trasnsaction destination state machine lo reflejará en el valor de la variable TransactionStatus_D.

Si no es necesario obtener información de otros nodos se pasará a comprobar si la respuesta está disponible. Si no lo está se lanzará un thread (thread 2, RetACK) que retransmitirá el ACK del mensaje recibido si se reciben copias del mensaje ya procesado (que no se volverá a procesar). Mientras el thread principal decrementará el contador de transacción hasta que la respuesta esté disponible. Si el contador llega a cero (tiempo de transacción acabado) y no se dispone de la respuesta, se informará que la transacción ha fallado mediante la variable TransactionStatus_D.

Si la respuesta está disponible, será enviada al nodo fuente de la petición y se pasará a esperar el ACK de dicha respuesta. Para ello se lanza un thread (thread 3, RX_ACK_D) que comprueba si se ha recibido el ACK del mensaje. Si lo que se recibe es una copia de la petición ya procesada se procederá a enviar el correspondiente ACK, y se pasará a esperar la recepción de otro mensaje. Si se recibe el ACK se informará al thread principal sobre la recepción del mismo y la transacción acabará correctamente. Mientras se espera recibir el ACK el thread principal comprueba el contador de transacción y el de retransmisión de tal forma que si el tiempo de transacción se ha acabado sin recibir ACK o se ha alcanzado el número máximo de retransmisiones sin recibir dicho ACK la transacción será errónea y así lo reflejará la variable TransactionStatus_D. Por el contrario, si no se ha acabado el tiempo de transacción ni el número de retransmisiones ha llegado a su máximo, se procederá a comprobar de nuevo si se ha recibido el ACK. Además, si el timer de retransmisión ha expirado se procederá a reenviar el mensaje de respuesta otra vez.

3.3 Implementación de primitivas

El esquema de implementación en C de cada una de las primitivas depende del tipo de éstas. Las primitivas pertenecientes al mismo tipo siguen el mismo esquema de implementación en C. Los distintos tipos de primitivas que intervienen el handover son los siguientes:

- Tipo HO Request: El diagrama de flujo que representa la implementación de este tipo de primitivas es el siguiente (Figura 3.3):

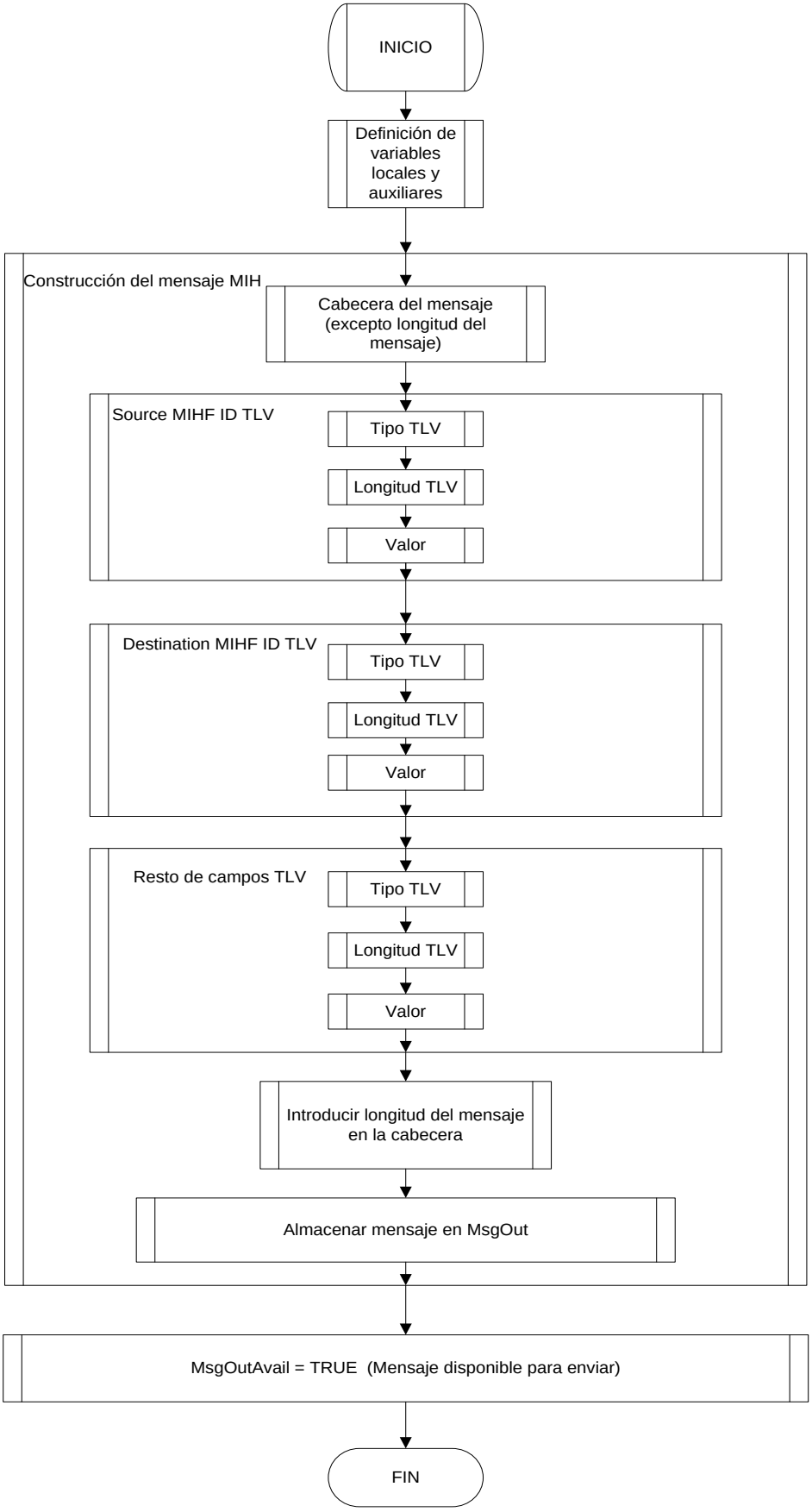

Figura 3.3: Diagrama de flujo de primitivas HO Request

Este tipo de primitivas se encargan de construir y guardar los mensajes MIH de peticiones. Cuando el mensaje está disponible, se cambia el valor de la variable MsgOutAvail a TRUE para indicar que es necesario iniciar una instancia de la transaction source state machine relativa a una nueva transacción.

Para la construcción de la cabecera se guardan los dos primeros bytes en variables auxiliares de tipo unsigned char (1 byte) con valores que dependen del tipo de mensaje a construir, y se copian en las primeras posiciones de la estructura donde se guarda el mensaje mediante la función memcpy. Los dos siguientes bytes (Message_ID) se almacenan temporalmente en una variable de tipo unsigned short (2 bytes) con el valor correspondiente al tipo de mensaje que se va a construir y se copian en la estructura de almacenamiento del mensaje a continuación de los dos bytes descritos anteriormente. Los dos siguientes bytes (Rsvd2 y Transaction_ID) se almacenan temporalmente en una variable de tipo unsigned short (2 bytes) y se copian a continuación del Message_ID en la estructura del mensaje. Los dos últimos bytes de la cabecera representan la longitud de los datos (Payload_Length) y su valor se introduce al final de la primitiva ya que es necesario calcular la longitud de dichos datos. Para reservar estos dos bytes se incrementa el puntero auxiliar que controla la posición de escritura de tal forma que se salta dos posiciones (bytes) de la estructura de almacenamiento provisional del mensaje (array de caracteres) y se sitúa en la dirección de comienzo de escritura de datos.

A continuación de la cabecera se guarda la estructura TLV que lleva el Source Identifier MIHF ID. Para ello se introduce el valor correspondiente al tipo de esta estructura en el primer byte de los datos. Los siguientes bytes representan la longitud de esta estructura y su codificación es la descrita en el apartado 2.4.3.2.1. El valor de la estructura se introduce a continuación de la longitud.

Después de la estructura TLV que lleva el Source Identifier MIHF ID se guarda la estructura TLV que lleva el Destination Identifier MIHF ID, y que se codifica igual que la anteriormente descrita estructura TLV Source Identifier MIHF ID.

La última parte de los mensajes es de longitud variable, es decir, consta de diferentes campos codificados en forma TLV según el tipo de mensaje y que se sitúan uno a continuación del otro. La forma de codificación de cada una de las estructuras TLV de las que consta el mensaje es igual a las descritas anteriormente, es decir, un byte para el tipo, los bytes necesarios para la longitud y por último los datos de la estructura.

A la vez que se van introduciendo cada una de las estructuras se va calculando la longitud de las mismas, de tal forma que al final se obtiene la longitud total en bytes del campo de datos del mensaje y dicho valor se introduce en los dos bytes de la cabecera reservados a tal efecto.

Para guardar el mensaje a enviar se reservan, de forma dinámica, los bytes necesarios para dicho mensaje y se copia el contenido de la estructura donde se guarda el mensaje de forma provisional a la memoria reservada cuyo comienzo viene señalado por la variable tipo puntero MsgOut.

Por último se pone a TRUE la variable MsgOutAvail para informar que hay un mensaje listo para ser enviado y por lo tanto es necesario iniciar una instancia de la transaction source state machine correspondiente a una nueva transacción.

- Tipo HO Response: El diagrama de flujo que representa la implementación de este tipo de primitivas es el siguiente (Figura 3.4):

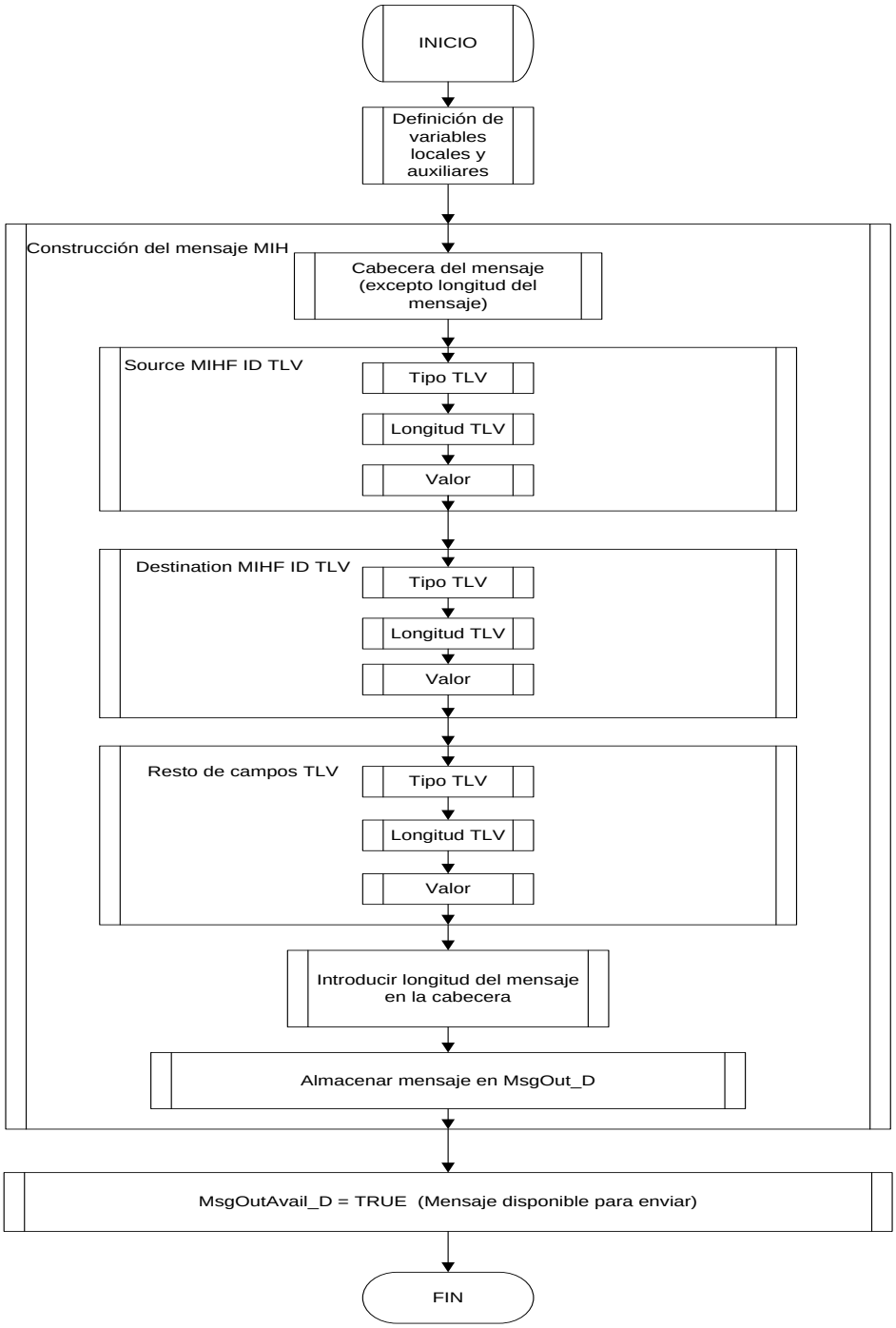

Figura 3.4: Diagrama de flujo de primitivas HO Response

Este tipo de primitivas se encargan de construir y guardar los mensajes MIH de respuesta a peticiones. Cuando se ha construido el mensaje, se cambia el valor de la

variable MsgOutAvail_D a TRUE para indicar que el mensaje de respuesta a una petición está disponible.

La construcción de cada uno de los campos del mensaje que este tipo de primitivas construye sigue la misma estructura que las anteriores primitivas descritas, es decir, las primitivas de tipo HO Request.

- Tipo HO Indication: El diagrama de flujo que representa la implementación de este tipo de primitivas es el siguiente (Figura 3.5):

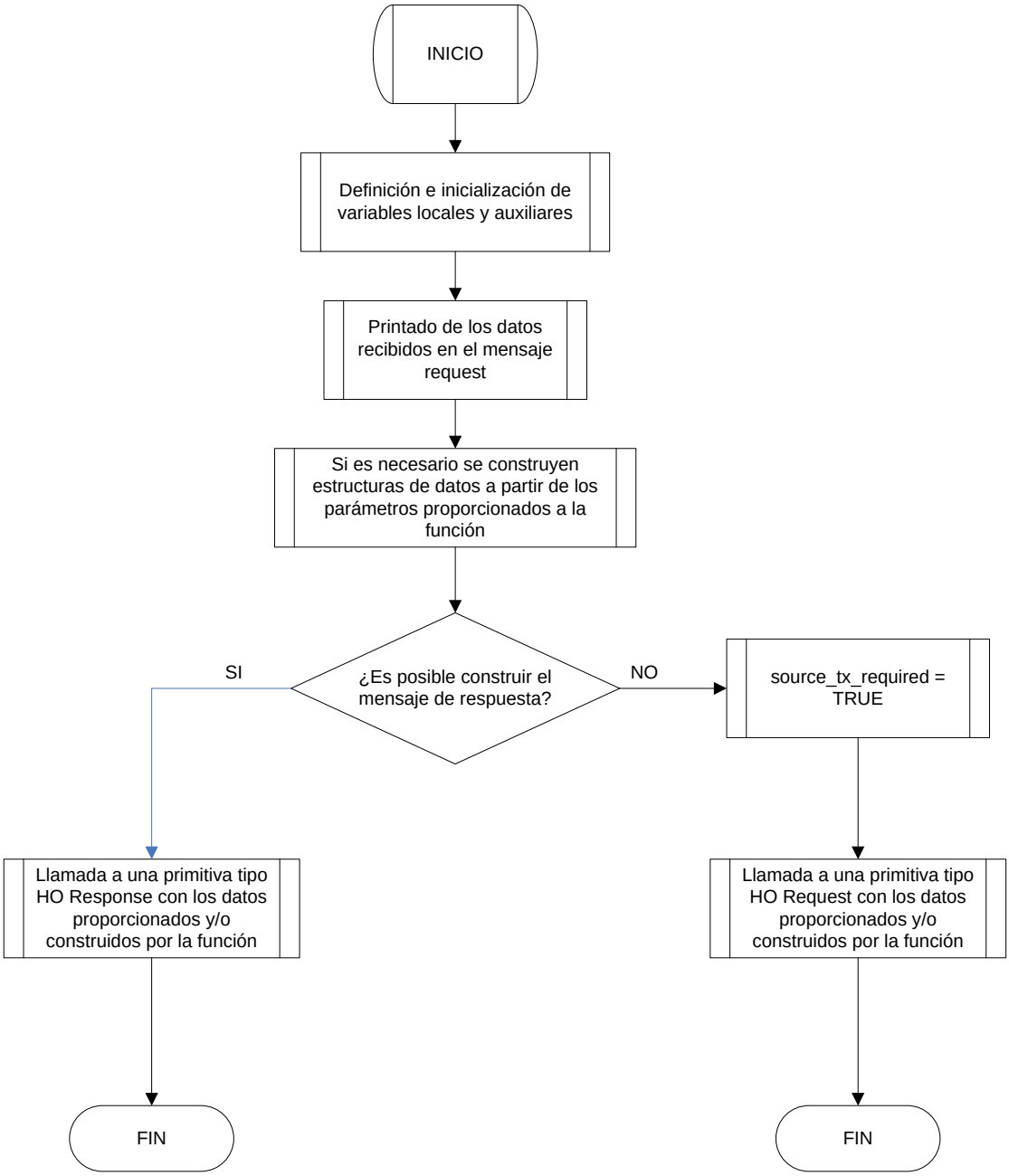

Figura 3.5: Diagrama de flujo de primitivas HO Indication

Este tipo de primitivas se encargan de mostrar por pantalla los datos recibidos en el mensaje request. Si es necesario se construyen y almacenan nuevas estructuras a partir de los datos pasados por parámetro a la primitiva que son necesarias para llamar a las primitivas de tipo HO. En ocasiones el mensaje de respuesta no es posible construirlo ya que es necesario consultar a otro nodo antes de poder obtener la respuesta; en este caso la variable source_tx_required se pone a TRUE para indicar a la transaction destination state machine que es necesario iniciar una instancia de la transacción source state machine para obtener los datos necesarios y poder construir la respuesta.

En el caso de que se pueda construir la respuesta sin la necesidad de obtener información de otros nodos se realiza una llamada a una primitiva de tipo HO Response que es la encargada de construir el mensaje MIH.

- Tipo HO Confirm: El diagrama de flujo que representa la implementación de este tipo de primitivas es el siguiente (Figura 3.6):

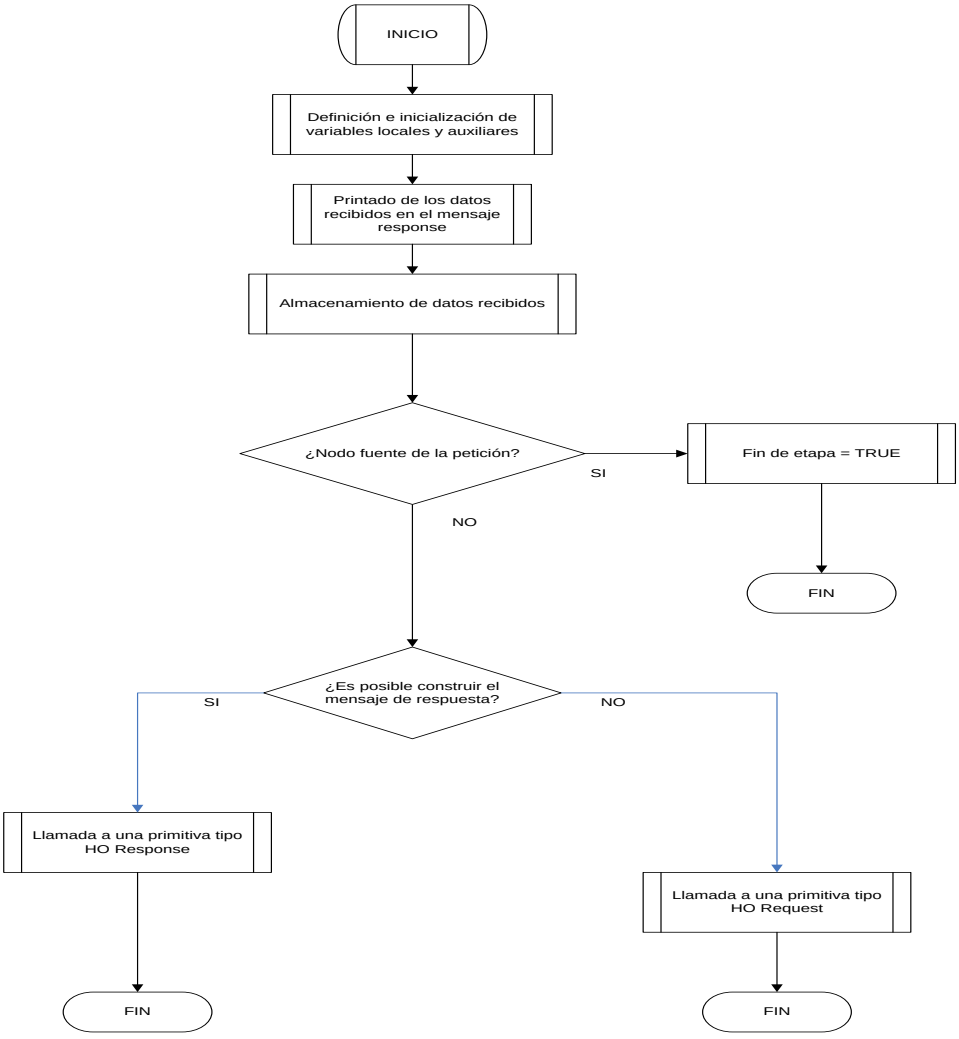

Figura 3.6: Diagrama de flujo de primitivas HO Confirm

Este tipo de primitivas se encargan de mostrar por pantalla los datos recibidos en el mensaje response y de almacenar los datos útiles para el handover recibidos en dicho mensaje. La llamada a esta primitiva por el nodo fuente de una petición, después de recibir el response, da como resultado la puesta a TRUE de la variable que indica el fin de una de las etapas del handover. Cuando no es posible construir una respuesta a una petición de forma inmediata y es necesario abrir una transacción para obtener información de otro nodo, esta primitiva se encarga de realizar una llamada a una primitiva de tipo request encargada de construir el mensaje de petición. Si por el contrario, la respuesta a la petición original es posible construirla con los datos después de recibir el mensaje response, esta primitiva se encarga de construir el mensaje de respuesta a la petición original a través de la llamada a una primitiva de tipo response que se encarga de construir la respuesta.

- Tipo Link Actions: Estas primitivas son usadas durante la fase de establecimiento de conexión a nivel 2 del handover. Se corresponden con las acciones llevadas a cabo para controlar los enlaces y ejecutar las acciones necesarias sobre ellos. Su implementación consiste en la simulación de dichas acciones a nivel de enlace, de tal forma que estas primitivas simulan de forma descriptiva el comportamiento del enlace, es decir, se encargan de mostrar por pantalla las acciones y resultados del comportamiento del enlace a nivel 2 (simulado) durante el handover.

3.4 Programas principales

3.4.1 Mobile Node

El diagrama de flujo de la implementación en C del programa principal que controla el handover en el Mobile Node es el representado en la siguiente figura (Figura 3.7):

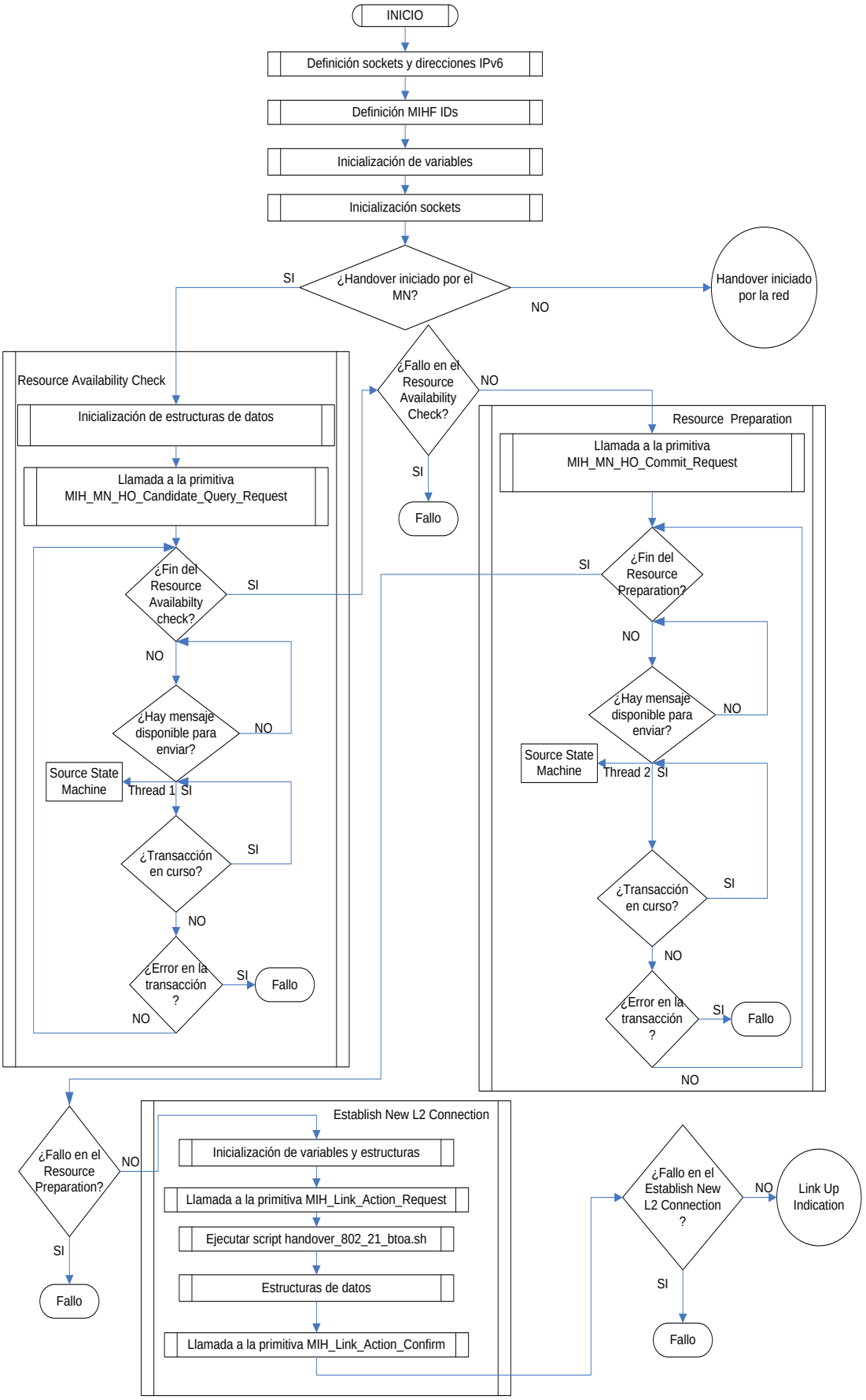

Figura 3.7: Diagrama de flujo Mobile_Node.c

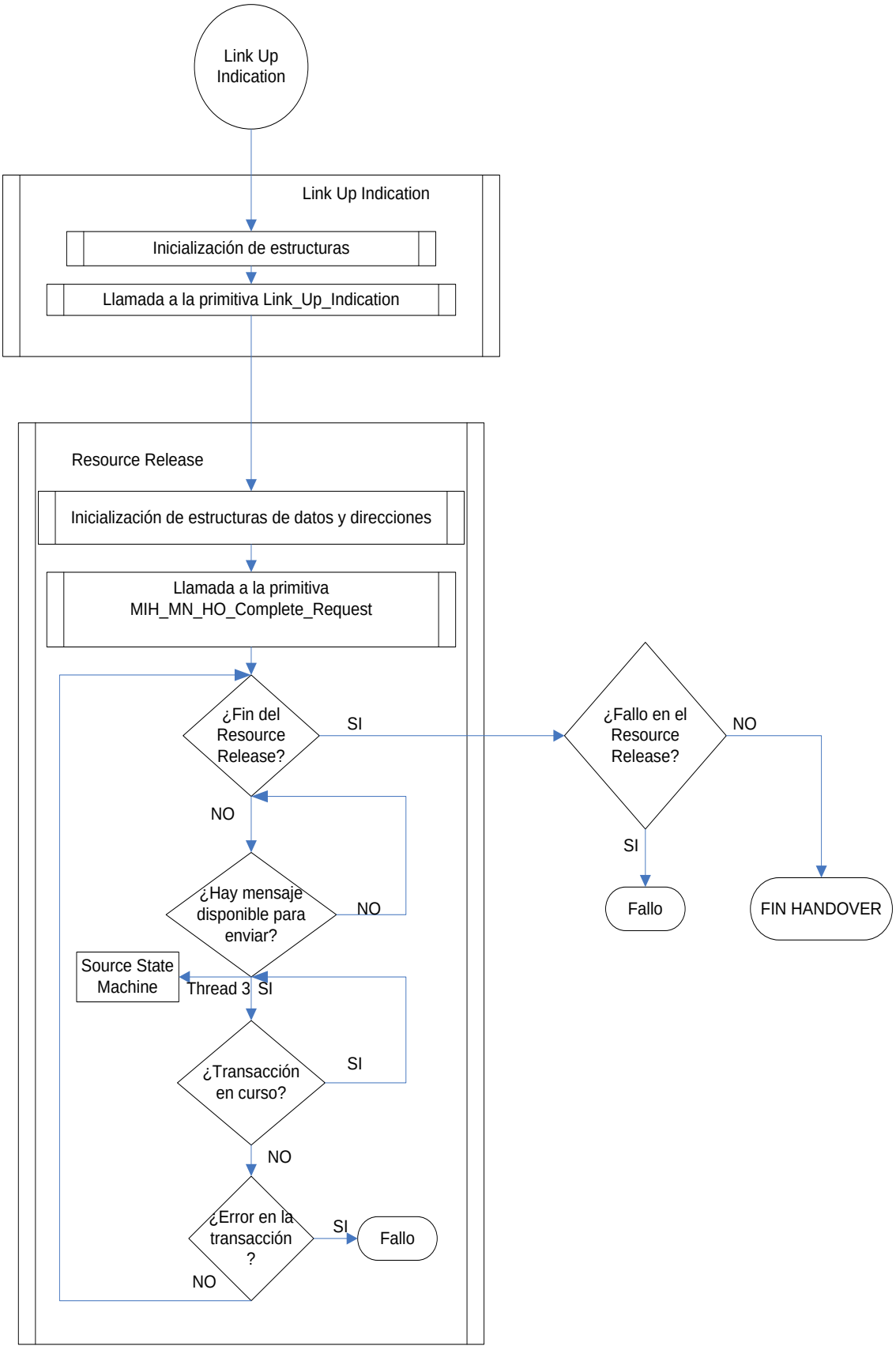

Figura 3.7 (Continuación 1): Diagrama de flujo Mobile_Node.c

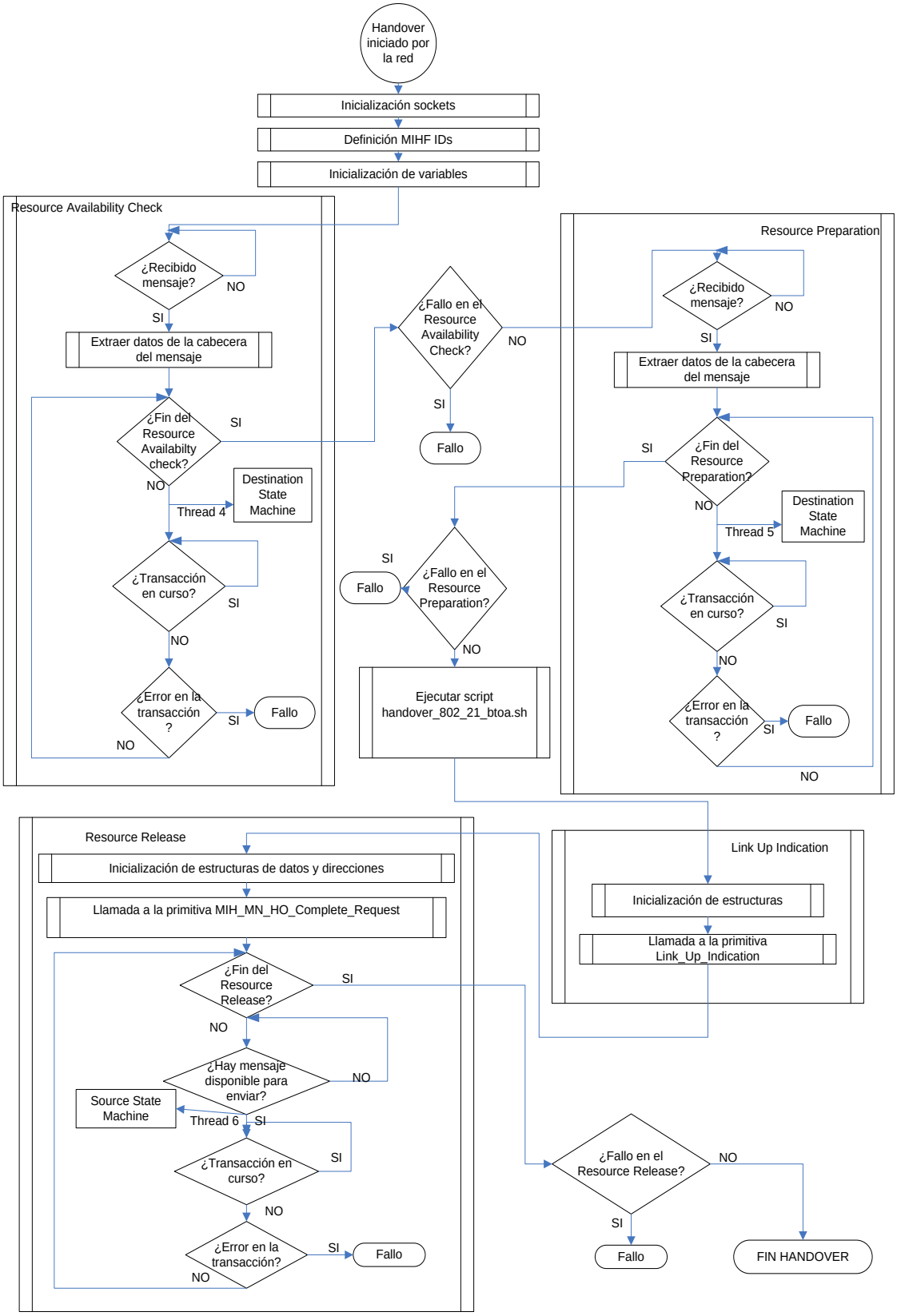

Figura 3.7 (Continuación 2): Diagrama de flujo Mobile_Node.c

El programa principal que controla el handover en el MN (Mobile_Node.c) realiza cada una de las fases de las que consta dicho handover, utilizando para ello las primitivas y mensajes descritos en el los apartados 3.5 y 3.6.

El programa empieza definiendo las variables, sockets y direcciones que se utilizan a lo largo del programa, así como inicializando las mismas.

A continuación se consulta la variable que indica si el handover es iniciado por el MN o por la Red.

- Handover iniciado por el MN:

Se inicia la fase Resource Availability Check, para comprobar la disponibilidad de los recursos existentes en las redes candidatas. Esta fase se inicia con la llamada a la primitiva MIH_MN_HO_Candidate_Query_Request. Dentro del bucle Resource Availability Check se comprueba si hay un mensaje listo para ser enviado hasta que éste está disponible. Una vez está disponible el mensaje se lanza un thread que se corresponde con la transaction source state machine de tal forma que se inicia una transacción. Cuando la transacción acaba se comprueba si ha habido algún error. Si es así, el handover es finalizado. Si no ha habido ningún error se pasa a la siguiente fase del handover.

La siguiente fase, Resource Preparation, se inicia con la llamada a la primitiva MIH_MN_HO_Commit_Request. En esta fase se preparan los recursos que son necesarios para realizar el handover. Dentro del bucle Resource Preparation se comprueba si hay un mensaje listo para ser enviado hasta que éste está disponible. Una vez está disponible el mensaje se lanza un thread que se corresponde con la transaction source state machine de tal forma que se inicia una transacción. Cuando la transacción acaba se comprueba si ha habido algún error. Si es así, el handover es finalizado. Si no ha habido ningún error se pasa a la siguiente fase del handover.

En la siguiente fase, Establish New L2 Connection, se preparan los enlaces fuente y destino que intervienen en el handover. Se inicia con la llamada a la primitiva MIH_Link_Action_Request. Posteriormente se ejecuta el script handover_802_21_btoa.sh, que realiza el handover mediante la secuencia de instrucciones descritas en el apartado 5.5.1. Por último se realiza la llamada a la primitiva MIH_Link_Action_Confirm. Se comprueba que no ha habido errores (si los ha habido se para el handover) y se pasa a la siguiente fase.

En la fase Link Up Indication, se informa que la conexión a nivel 2 ha sido establecida. Para ello se realiza la llamada a la primitiva Link_Up_Indication.

La última fase es la Resource Release, y en esta fase se liberan los recursos que estaban siendo utilizados antes del handover y se informa del resultado del mismo. Dentro del bucle Resource Release se comprueba si hay un mensaje listo para ser enviado hasta que éste está disponible. Una vez está disponible el mensaje se lanza un thread que se corresponde con la transaction source state machine de tal forma que se inicia una transacción. Cuando la transacción acaba se comprueba si ha habido algún error. Si es así, el handover acaba de forma errónea y no se da por bueno. Si por el contrario no ha habido ningún error el handover termina de forma satisfactoria.

- Handover iniciado por la Red:

Se inicia la fase Resource Availability Check, para comprobar la disponibilidad de los recursos existentes en las redes candidatas. En este caso es la red la que inicia el handover, por lo tanto el MN inicia esta fase esperando recibir un mensaje de comienzo del mismo. Una vez recibido el mensaje se extraen los datos de la cabecera y se lanza un thread que se corresponde con la destination source state machine. Se espera a que acabe la transacción y se comprueba el resultado de la misma. Si ha habido un error se detiene el handover. Si no es así se pasa a la siguiente fase.

La siguiente fase es la Resource Preparation. En esta fase se preparan los recursos que son necesarios para realizar el handover. El MN se queda a la espera de recibir el mensaje de la Serving Network. Una vez que lo recibe se lanza un thread de la destination source state machine. Se espera a que acabe la transacción y se comprueba el resultado de la misma. Si ha habido un error se detiene el handover. Si no es así se pasa a la siguiente fase.

En la fase Link Up Indication, se informa que la conexión a nivel 2 ha sido establecida. Para ello se realiza la llamada a la primitiva Link_Up_Indication.

La última fase es la Resource Release, y en esta fase se liberan los recursos que estaban siendo utilizados antes del handover y se informa del resultado del mismo. Su implementación es la misma que en el caso del handover iniciado por el MN.

3.4.2 Serving Network

El diagrama de flujo de la implementación en C del programa principal que controla el handover en la Serving Network es el representado en la siguiente figura (Figura 3.8):

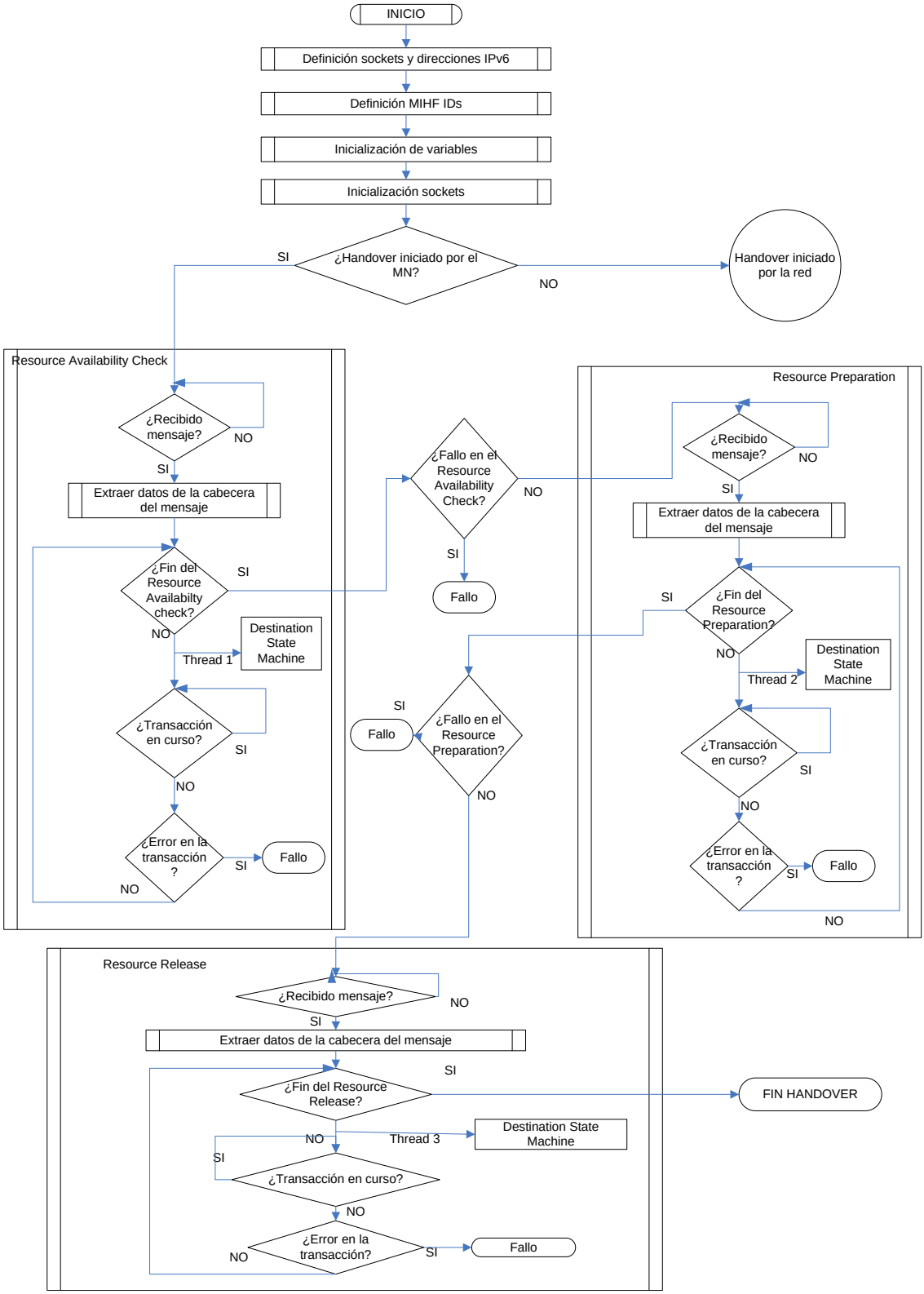

Figura 3.8: Diagrama de flujo Serving_Network.c

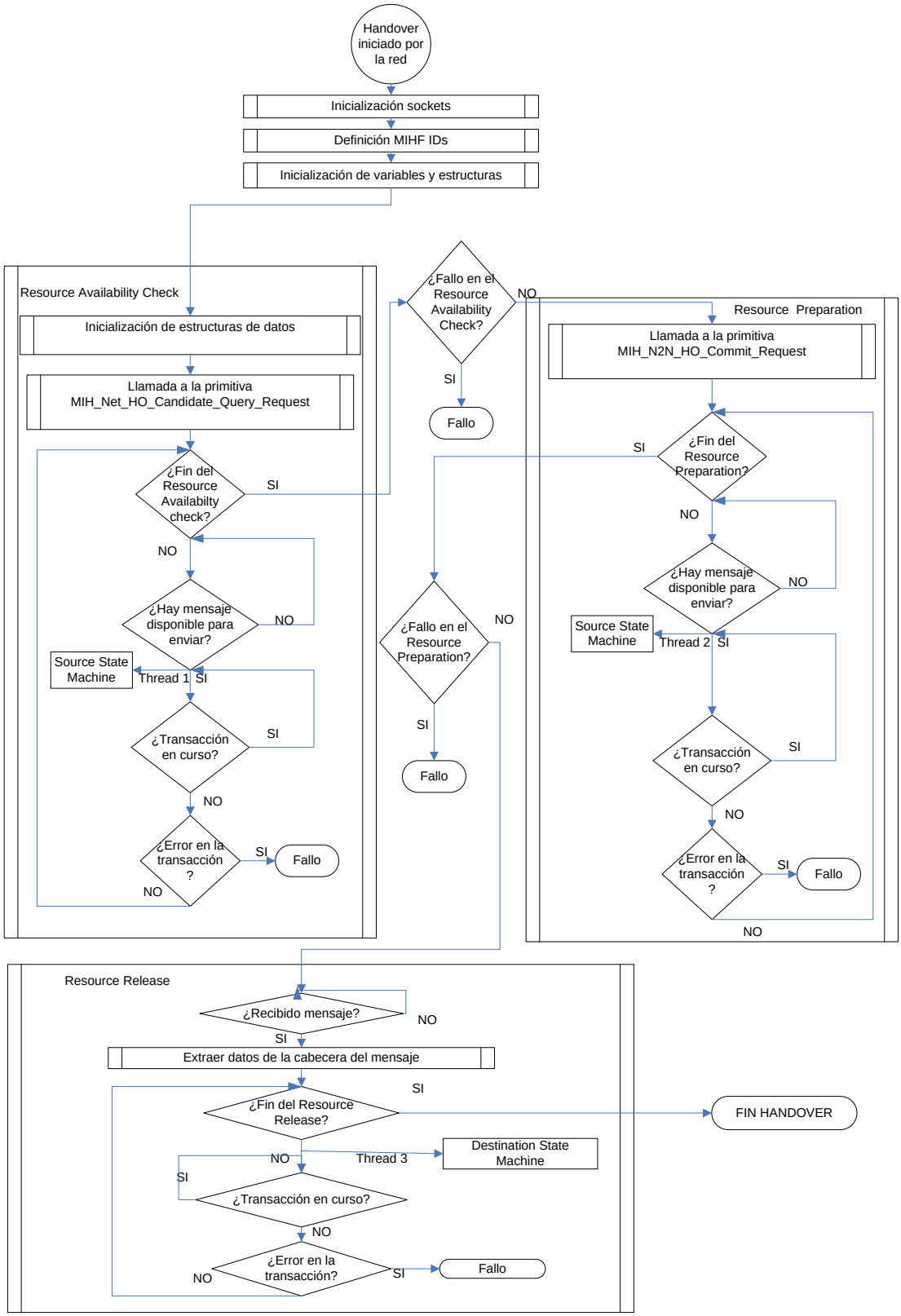

Figura 3.8 (Continuación): Diagrama de flujo Serving_Network.c

El programa principal que controla el handover en la Serving Network (Serving_network.c) realiza cada una de las fases de las que consta dicho handover, utilizando para ello las primitivas y mensajes descritos en el los apartados 3.5 y 3.6.

El programa empieza definiendo las variables, sockets y direcciones que se utilizan a lo largo del programa, así como inicializando las mismas.

A continuación se consulta la variable que indica si el handover es iniciado por el MN o por la Red.

- Handover iniciado por el MN:

Se inicia la fase Resource Availability Check, para comprobar la disponibilidad de los recursos existentes en las redes candidatas. La implementación de esta fase es la misma que la descrita en el apartado 3.5.1 para el handover iniciado por la Red, donde el receptor del mensaje es la Serving Network.

La siguiente fase es la Resource Preparation. En esta fase se preparan los recursos que son necesarios para realizar el handover. La implementación de esta fase es la misma que la descrita en el apartado 3.5.1 para el handover iniciado por la Red, donde el receptor del mensaje es la Serving Network.

La última fase es la Resource Release, y en esta fase se liberan los recursos que estaban siendo utilizados antes del handover y se informa del resultado del mismo. La Serving Network se queda a la espera de recibir el mensaje de la red candidata. Una vez que lo recibe se lanza un thread de la destination source state machine. Se espera a que acabe la transacción y se comprueba el resultado de la misma. Si es así, el handover acaba de forma errónea y no se da por bueno. Si por el contrario no ha habido ningún error el handover termina de forma satisfactoria.

- Handover iniciado por la Red:

Se inicia la fase Resource Availability Check, para comprobar la disponibilidad de los recursos existentes en las redes candidatas. Esta fase se inicia con la llamada a la primitiva MIH_Net_HO_Candidate_Query_Request. Dentro del bucle Resource Availability Check se comprueba si hay un mensaje listo para ser enviado hasta que éste está disponible. Una vez está disponible el mensaje se lanza un thread que se corresponde con la transaction source state machine de tal forma que se inicia una transacción. Cuando la transacción acaba se comprueba si ha habido algún error. Si es así, el handover es finalizado. Si no ha habido ningún error se pasa a la siguiente fase del handover.

La siguiente fase, Resource Preparation, se inicia con la llamada a la primitiva MIH_N2N_HO_Commit_Request. En esta fase se preparan los recursos que son necesarios para realizar el handover. Dentro del bucle Resource Preparation se comprueba si hay un mensaje listo para ser enviado hasta que éste está disponible. Una vez está disponible el mensaje se lanza un thread que se corresponde con la transaction source state machine de tal forma que se inicia una transacción. Cuando la transacción acaba se comprueba si ha habido algún error. Si es así, el handover es finalizado. Si no ha habido ningún error se pasa a la siguiente fase del handover.

La última fase es la Resource Release, y en esta fase se liberan los recursos que estaban siendo utilizados antes del handover y se informa del resultado del mismo. La implementación de esta fase es la misma que la descrita en el caso del handover iniciado por el MN de este apartado.

3.4.3 Candidate Networks

El diagrama de flujo de la implementación en C del programa principal que controla el handover en la Candidate Nework 1 es el representado en la siguiente figura (Figura 3.9):

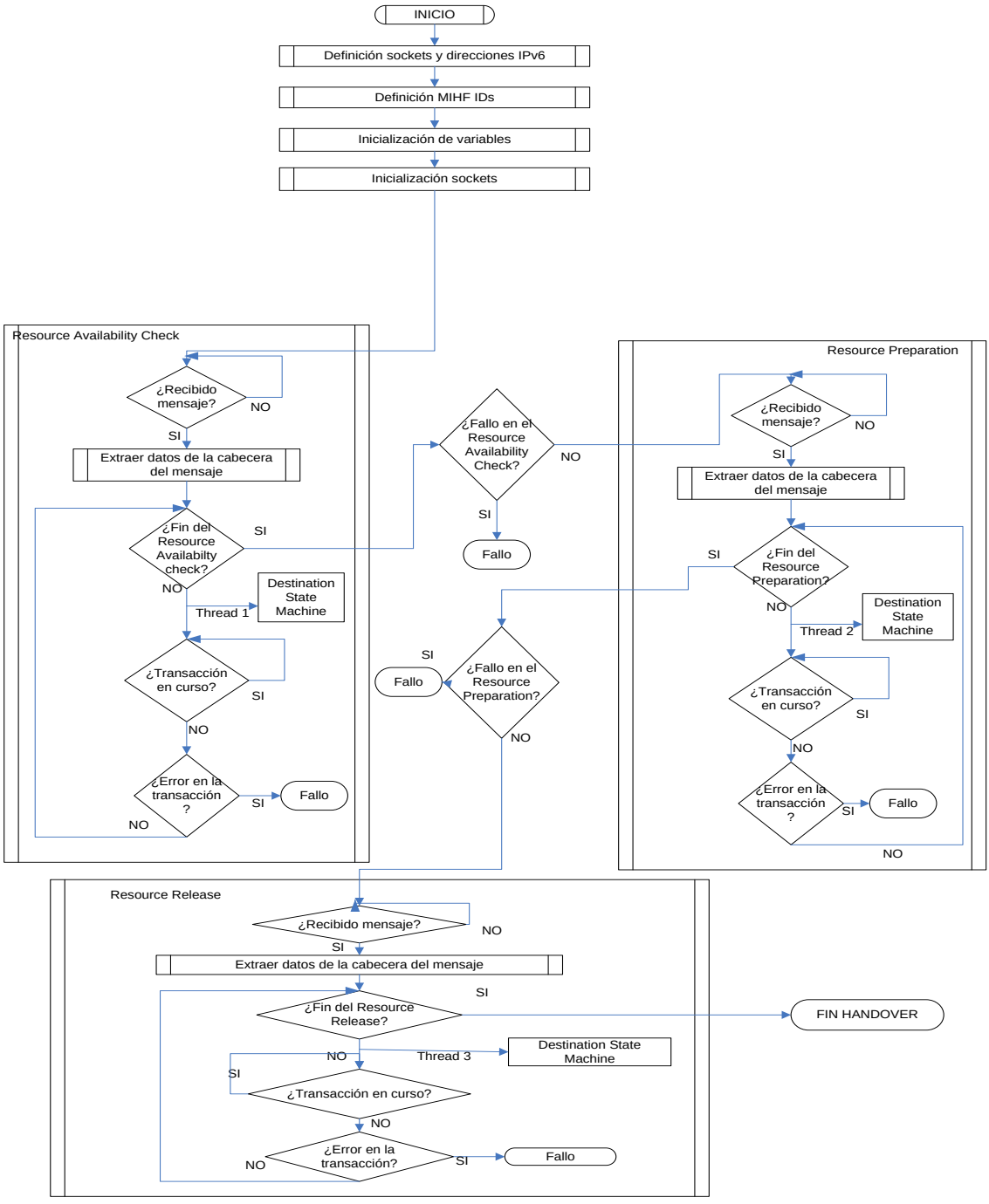

Figura 3.9: Diagrama de flujo Candidate1.c

El programa principal que controla el handover en la Candidate Network 1 (Candidate1.c) realiza cada una de las fases de las que consta dicho handover, utilizando para ello las primitivas y mensajes descritos en el los apartados 3.5 y 3.6.

El programa empieza definiendo las variables, sockets y direcciones que se utilizan a lo largo del programa, así como inicializando las mismas.

Las fases de las que consta son Resource Availability Check, Resource Preparation y Resource Release, y su implementación es la misma que la descrita en el handover iniciado por el MN del apartado 3.4.2

El diagrama de flujo de la implementación en C del programa principal que controla el handover en la Candidate Nework 2 es el representado en la siguiente figura (Figura 3.10):

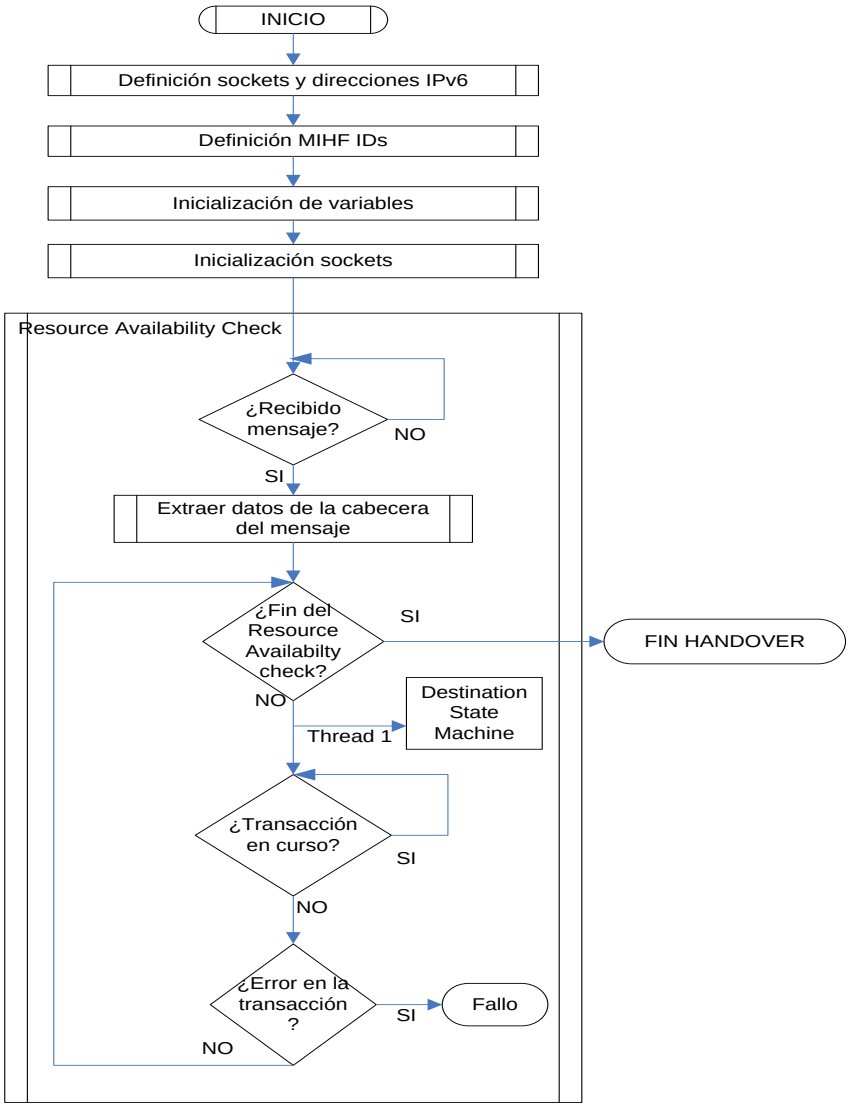

Figura 3.10: Diagrama de flujo Candidate2.c

El programa principal que controla el handover en la Candidate Network 2 (Candidate2.c) realiza cada una de las fases de las que consta dicho handover, utilizando para ello las primitivas y mensajes descritos en el los apartados 3.5 y 3.6.

El programa empieza definiendo las variables, sockets y direcciones que se utilizan a lo largo del programa, así como inicializando las mismas.

Sólo consta de la fase Resource Availability Check y su implementación es la misma que la descrita en el handover iniciado por el MN del apartado 3.4.2

3.5 Handover iniciado por el Mobile Node

El procedimiento para el handover iniciado por el Mobile Node se muestra en la siguiente figura (Figura 3.11):

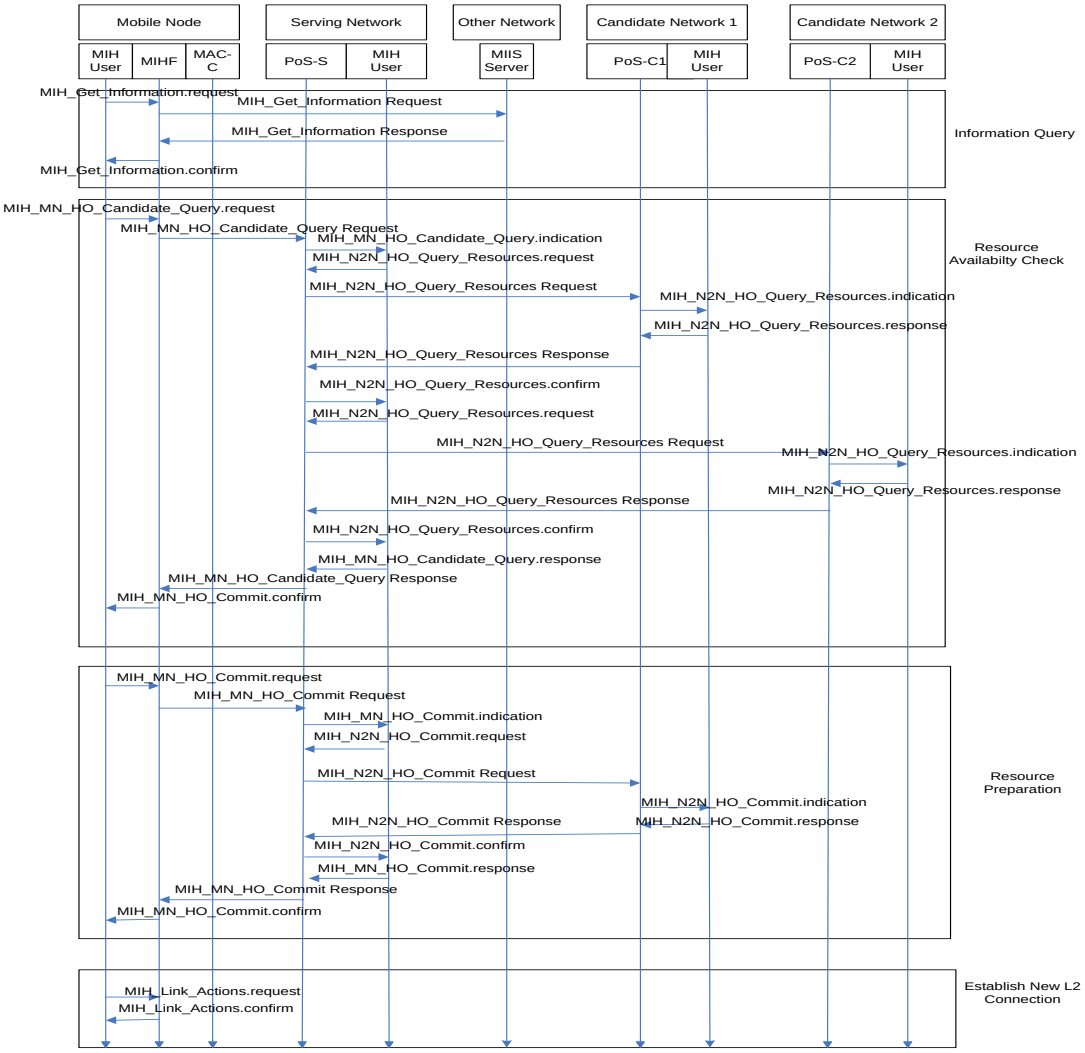

Figura 3.11: Handover iniciado por el Mobile Node

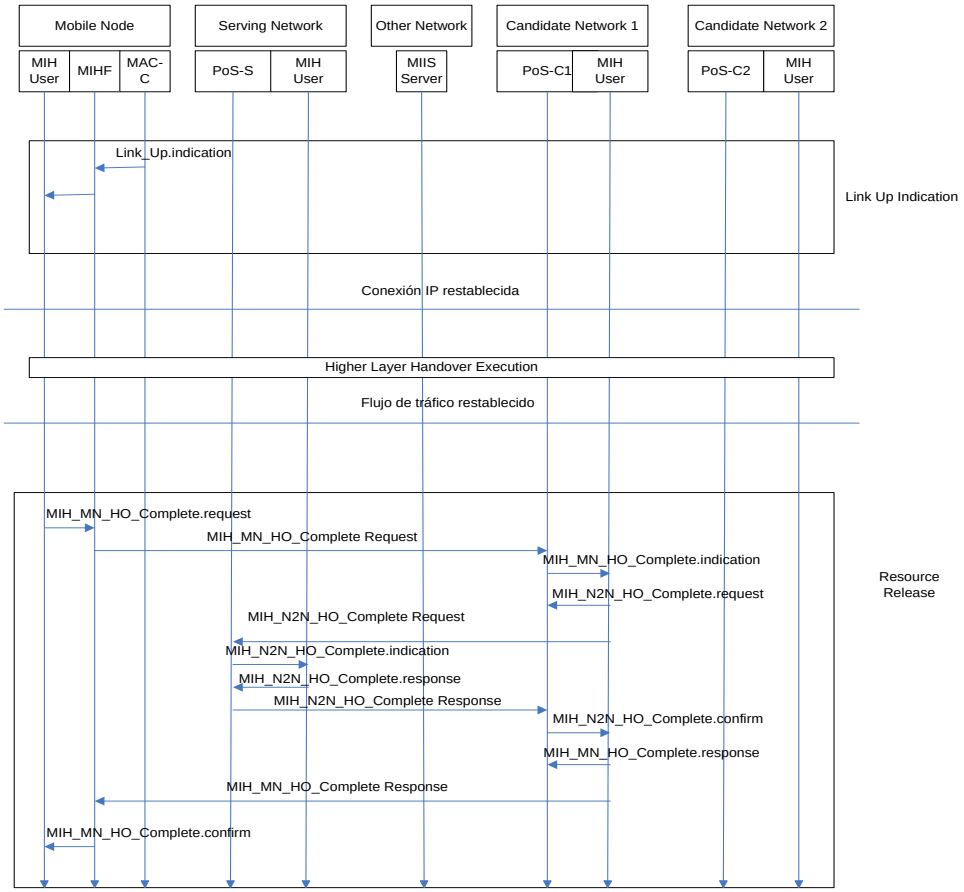

Figura 3.11 (Continuación): Handover iniciado por el Mobile Node

El MN solicita información al MIH Information Server (MIIS) acerca de sus redes vecinas mediante el mensaje MIH_Get_Information Request. El MIIS responde mediante el mensaje MIH_Get_Information Response con la información disponible.

El MN empieza la primera fase del handover (Resource Availability Check) mediante el envío del mensaje MIH_MN_HO_Candidate Query Request a la Serving Network solicitando la disponibilidad de recursos en las posibles redes candidatas al handover. La Serving Network solicita dicha información a cada una de las redes candidatas mediante el envío del mensaje MIH_N2N_HO_Query _Resorces Request. Cada una de estas redes candidatas responde con la información solicitada mediante el mensaje MIH_N2N_HO_Query _Resorces Response. Cuando la Serving Network tiene la información necesaria construye el mensaje MIH_MN_HO_Candidate Query Response en respuesta a la petición inicial del MN.

La segunda fase del handover (Resource Preparation) comienza con el envío, por parte del MN, del mensaje MIH_MN_HO_Commit Request solicitando la reserva de los recursos en la red elegida para realizar el handover. Cuando la Serving Network recibe este mensaje realiza la petición de reserva de recursos para el handover a la red donde se va a realizar el mismo mediante el mensaje MIH_N2N_HO_Commit Request. Dicha red candidata le responde con la información de los recursos mediante el mensaje MIH_N2N_HO_Commit Response. A partir de los datos recibidos la Serving Network construye el mensaje de respuesta (MIH_MN_HO_Commit Response) que envía al MN con la información obtenida.

En las siguientes fases (Establish L2 Connection y Link Up Indication) se lleva a cabo las acciones sobre los enlaces a nivel 2, como, por ejemplo, la desactivación del antiguo enlace y la activación del nuevo.

Una vez que se ha reestablecido la conectividad IP y el flujo de datos de las capas superiores, se inicia la última fase del handover (Resource Release) . Esta fase se inicia con el envío del mensaje MIH_MN_HO_Complete Request por parte del MN a la red donde se ha realizado el handover para informar del resultado del mismo. La red candidata informa a la antigua Serving Network del resultado del handover y le envía una petición mediante el mensaje MIH_N2N_HO_Complete Request para la liberación de los recursos que estaba usando el MN en dicha red. La información del estado de liberación de los recursos es enviada al MN mediante el mensaje MIH_MN_HO_Complete Response por la red candidata, que ha pasado ha ser la nueva Serving Network.

3.5.1 Primitivas

3.5.1.1 Mobile Node

A continuación se describen las primitivas usadas por el MN durante el proceso de handover iniciado por dicho nodo y mostrado en la Figura 3.11.

● ***MIH_MN_HO_Candidate_Query_Request***

Esta primitiva es usada por los usuarios MIH para informar a la MIHF del MN que es necesario realizar una petición solicitando información de los posibles candidatos a un handover.

Se encarga de construir el mensaje de petición(MIH_MN_HO_Candidate_Query Request) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Request descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: Identificador de la MIHF remota que es el destinatario del mensaje MIH

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ CandidateLinkList: Lista de PoAs candidatos para realizar el handover, donde el primer elemento es el de mayor preferencia.

Tipo de dato estándar IEEE 802.21: LIST(LINK_POA_LIST)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la CandidateLinkList.

→ QoSResourceRequirements: QoS para los recursos mínima en la red candidata

Tipo de dato estándar IEEE 802.21: QOS_LIST
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena los QoSResourceRequirements.

→IPConfigurationMethods (Opcional): Métodos de configuración IP actuales

Tipo de dato estándar IEEE 802.21: IP_CFG_MTHDS
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena los IPConfigurationMethods. Su valor es NULL cuando no es utilizado.

→ DHCPServerAddress (Opcional): Dirección IP del servidor DHCP actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la DHCPServerAddress. Su valor es NULL cuando no es utilizado.

→ FAAdresss (Opcional): Dirección IP del servidor Foreign Agent actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la FAAdress. Su valor es NULL cuando no es utilizado.

→ AccessRouterAdresss (Opcional): Dirección IP del router de acceso actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el AccessRouterAdress. Su valor es NULL cuando no es utilizado.

● ***MIH_MN_HO_Candidate_Query_Confirm***

Esta primitiva es usada para informar a los usuarios MIH que se ha recibido la respuesta (mensaje MIH_MN_HO_Candidate Query Response) a la petición de información sobre la redes candidatas al handover.

Se encarga de modificar el valor de la variable Resource_Availability_Check, poniéndola a TRUE, para indicar que se ha llegado al final de la fase Resource Availability Check del handover. Su estructura C se corresponde con el tipo HO Confirm descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: Identificador de la MIHF remota que invoca esta primitiva

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ PreferredCandidateLinkList: Lista cuyos elementos son datos son del tipo RQ_RESULT que contienen las redes que deberían considerarse para realizar el handover y que pueden ser diferentes a las proporcionadas en la petición; y donde el primer elemento es el de mayor preferencia.

Tipo de dato estándar IEEE 802.21: LIST(RQ_RESULT)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la PreferredCandidateLinkList.

● ***MIH_MN_HO_Commit_Request***

Esta primitiva es usada para informar a la Serving Network de cual ha sido la red elegida para realizar el handover.

Se encarga de construir el mensaje de petición (MIH_MN_HO_Commit Request) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Request descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: MIHF ID de la Serving Network que es el objetivo de esta primitiva

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ LinkType: Tipo de enlace de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: LINK_TYPE
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ TargetNetworkInfo: Información de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: TGT_NET_INFO

Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la TargetNetworkInfo.

● ***MIH_MN_HO_Commit_Confirm***

Esta primitiva es generada por la MIHF del MN para confirmar que se ha recibido el mensaje MIH_MN_HO_Commit Response enviado por la entidad MIHF de la Serving Network

Se encarga de modificar el valor de la variable Resource Preparation, cambiando su valor a TRUE, para indicar que se ha llegado al final de la fase Resource Preparation del handover. Su estructura C se corresponde con el tipo HO Confirm descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: MIHF ID de la serving network que envió el mensaje MIH_MN_HO_Commit Response.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→LinkType: Tipo de enlace de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: LINK_TYPE
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ TargetNetworkInfo: Información de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: TGT_NET_INFO
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la TargetNetworkInfo.

● ***Link_Action_Request***

Esta primitiva es usada por la MIHF para realizar la petición de ejecutar una acción en una conexión a nivel de enlace que posibilita una óptima gestión del mismo.

Se encarga de simular la ejecución de la lista de acciones pasadas por parámetro. Su estructura C se corresponde con el tipo Link Actions descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ LinkAction: Acción a realizar.

Tipo de dato estándar IEEE 802.21: LINK_ACTION
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la LinkAction.

→ ExecutionDelay: Tiempo, en ms, que se ha de esperar antes de realizar la acción. El valor 0 indica que la acción se debe llevar a cabo inmediatamente

Tipo de dato estándar IEEE 802.21: UNSIGNED_INT(2)
Tipo de dato C: unsigned short. Numero entero sin signo de longitud 2 bytes.

→ PoALinkAddress (Opcional): Dirección de enlace donde se han de reenviar los datos. Se usa cuando se ha de ejecutar la acción DATA_FWD_REQ

Tipo de dato estándar IEEE 802.21: LINK_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la PoALinkAddress. Su valor es NULL cuando no es utilizado.

● ***MIH_Link_Actions_Request***

Esta primitiva es usada para controlar (simular en este caso) el comportamiento de los enlaces.

Se encarga de construir los parámetros necesarios, a partir de los recibidos, que se utilizan en la llamada a la primitiva Link_Action_Request (descrita anteriormente), para simular las acciones realizadas sobre el enlace. Su estructura C se corresponde con el tipo Link Actions descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→DestinationIdentifier: Identifica la MIHF local o remota que será el destino de la petición.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ LinkActionList: Lista de acciones a realizar sobre el enlace.

Tipo de dato estándar IEEE 802.21: LIST(LINK_ACTION_REQ)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la LinkActionList.

● ***MIH_Link_Up_Indication***

Esta primitiva es usada para notificar un evento a la MIHF local.

Se encarga de simular la notificación de un evento. Su estructura C se corresponde con el tipo Link Actions descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→SourceIdentifier: Identificador de la MIHF que invoca esta primitiva

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ LinkIdentifier: Identificador del enlace asociado al evento

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el LinkIdentifier.

→ OldAccessRouter (Opcional): Dirección de enlace del router antiguo

Tipo de dato estándar IEEE 802.21: LINK_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la OldAccessRouter. Su valor es NULL cuando no es utilizado.

→ NewAccessRouter (Opcional): Dirección de enlace del router nuevo

Tipo de dato estándar IEEE 802.21: LINK_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la NewAccessRouter. Su valor es NULL cuando no es utilizado.

→ IPRenewalFlag (Opcional): Indica si el MN tiene que cambiar su IP en el nuevo PoA

Tipo de dato estándar IEEE 802.21: IP_RENEWAL_FLAG
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el IPRenewalFlag. Su valor es NULL cuando no es utilizado.

→ MobilityManagementSupport (Opcional): Indica el protocolo de movilidad soportado por el nuevo PoA.

Tipo de dato estándar IEEE 802.21: IP_RENEWAL_FLAG
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MobilityManagementSupport. Su valor es NULL cuando no es utilizado.

● ***Link_Up_Indication***

Esta primitiva es usada para notificar que una conexión a nivel 2 ha sido establecida en el enlace especificado.

Se encarga de realizar la llamada a la primitiva MIH_Link_Up_Indication para notificar el evento. Su estructura C se corresponde con el tipo Link Actions descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ LinkIdentifier: Identificador del enlace asociado al evento

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el LinkIdentifier.

→ OldAccessRouter (Opcional): Dirección de enlace del router antiguo

Tipo de dato estándar IEEE 802.21: LINK_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la OldAccessRouter. Su valor es NULL cuando no es utilizado.

→NewAccessRouter (Opcional): Dirección de enlace del router nuevo

Tipo de dato estándar IEEE 802.21: LINK_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la NewAccessRouter. Su valor es NULL cuando no es utilizado.

→ IPRenewalFlag (Opcional): Indica si el MN tiene que cambiar su IP en el nuevo PoA

Tipo de dato estándar IEEE 802.21: IP_RENEWAL_FLAG
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el IPRenewalFlag. Su valor es NULL cuando no es utilizado.

→MobilityManagementSupport (Opcional): Indica el protocolo de movilidad soportado por el nuevo PoA.

Tipo de dato estándar IEEE 802.21: IP_RENEWAL_FLAG
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MobilityManagementSupport. Su valor es NULL cuando no es utilizado.

● ***MIH_MN_HO_Complete_Request***

Esta primitiva es usada por los usuarios MIH para indicar que se ha completado el handover.

Se encarga de construir el mensaje de petición (MIH_MN_HO_Complete Request) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Request descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: Identificador de la MIHF remota que es el destinatario del mensaje MIH

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ TargetLinkIdentifier: Enlace destino del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

→ HandoverResult: Resultado del handover

Tipo de dato estándar IEEE 802.21: HO_RESULT
Tipo de dato C: unsigned char. Un byte de almacenamiento.

● ***MIH_MN_HO_Complete_Confirm***

Esta primitiva es usada por la MIHF para informar a los usuarios MIH que se ha recibido el mensaje MIH_MN_HO_Complete Response.

Se encarga de modificar el valor de la variable Resource_Release, poniéndola a TRUE, para indicar que se ha llegado al final de la fase Resource Release del handover. Su estructura C se corresponde con el tipo HO Confirm descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: MIHF ID del nodo que ha enviado el MIH_MN_HO_Complete Response.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ TargetLinkIdentifier: Enlace destino del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

3.5.1.2 Serving Network

A continuación se describen las primitivas usadas por la Serving Network durante el proceso de handover iniciado por el MN y mostrado en la Figura 3.11.

● ***MIH_MN_HO_Candidate_Query_Indication***

Esta primitiva es usada por la MIHF para indicar que se ha recibido el mensaje MIH_MN_HO_Candidate_Query Request desde el MN.

Se encarga de realizar la llamada a la primitiva MIH_N2N_HO_Query_Resorces_Request para solicitar información a las redes candidatas. Su estructura C se corresponde con el tipo HO Indication descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: MIHF remota que es la que invoca la primitiva

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ CandidateLinkList: Lista de PoAs candidatos para realizar el handover, donde el primer elemento es el de mayor preferencia.

Tipo de dato estándar IEEE 802.21: LIST(LINK_POA_LIST)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la CandidateLinkList.

→ QoSResourceRequirements: QoS para los recursos mínima en la red candidata

Tipo de dato estándar IEEE 802.21: QOS_LIST
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena los QoSResourceRequirements.

→ IPConfigurationMethods (Opcional): Métodos de configuración IP actuales

Tipo de dato estándar IEEE 802.21: IP_CFG_MTHDS
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena los IPConfigurationMethods. Su valor es NULL cuando no es utilizado.

→ DHCPServerAddress (Opcional): Dirección IP del servidor DHCP actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la DHCPServerAddress. Su valor es NULL cuando no es utilizado.

→ FAAdresss (Opcional): Dirección IP del servidor Foreign Agent actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la FAAdress. Su valor es NULL cuando no es utilizado.

→ AccessRouterAdresss (Opcional): Dirección IP del router de acceso actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el AccessRouterAdress. Su valor es NULL cuando no es utilizado.

● ***MIH_N2N_HO_Query_Resources_Request***

Esta primitiva es usada por la MIHF de la Serving Network para solicitar información sobre los recursos disponibles en las redes candidatas e información IP sobre las mismas.

Se encarga de construir el mensaje de petición MIH_N2N_Query_Resources Request) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Request descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: Identificador de la MIHF remota destino de la petición

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ QoSResourceRequirements: QoS para los recursos mínima en la red candidata

Tipo de dato estándar IEEE 802.21: QOS_LIST
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena los QoSResourceRequirements.

→ IPConfigurationMethods (Opcional): Métodos de configuración IP actuales

Tipo de dato estándar IEEE 802.21: IP_CFG_MTHDS
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena los IPConfigurationMethods. Su valor es NULL cuando no es utilizado.

→ DHCPServerAddress (Opcional): Dirección IP del servidor DHCP actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la DHCPServerAddress. Su valor es NULL cuando no es utilizado.

→ FAAdresss (Opcional): Dirección IP del servidor Foreign Agent actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la FAAdress. Su valor es NULL cuando no es utilizado.

→ AccessRouterAdresss (Opcional): Dirección IP del router de acceso actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el AccessRouterAdress. Su valor es NULL cuando no es utilizado.

→ CandidateLinkList: Lista de enlaces candidatos (APs o BSs) de una determinada red candidata. Cada enlace se identifica por su tipo y por la dirección de enlace del PoA

Tipo de dato estándar IEEE 802.21: LIST(LINK_ID)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la CandidateLinkList.

● ***MIH_N2N_HO_Query_Resources_Confirm***

Esta primitiva es usada por la MIHF de la Serving Network para informar de la recepción de un mensaje MIH_N2N_HO_Query_Resources Response.

Si el mensaje ha sido enviado por la primera red candidata, esta primitiva se encarga de realizar una llamada a la primitiva MIH_N2N_HO_Query_Resources_Request para solicitar información a la segunda red candidata. Si el mensaje ha sido enviado por la segunda red candidata, esta primitiva se encarga de llamar a la primitiva MIH_MN_HO_Candidate_Query_Response para construir el mensaje de respuesta a la petición del MN.

PARÁMETROS DE ENTRADA:

→SourceIdentifier: MIHF ID del nodo que ha enviado el mensaje MIH_N2N_Candidate_Query Response

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ ResourceStatus: Especifica si los recursos pedidos están disponibles en el nuevo PoA.

Tipo de dato estándar IEEE 802.21: LINK_RES_STATUS
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el ResourceStatus.

● ***MIH_MN_HO_Candidate_Query_Response***

Esta primitiva es usada por los usuarios MIH para informar a la MIHF del resultado de una petición.

Se encarga de construir el mensaje MIH_MN_HO_Candidate_Query Response (con los datos obtenidos de las redes candidatas) como respuesta al mensaje de petición del MN. Su estructura C se corresponde con el tipo HO Response descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: Identificador de la MIHF remota destino de la respuesta

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ PreferredCandidateLinkList: Lista cuyos elementos son datos son del tipo RQ_RESULT que contienen las redes que deberían considerarse para realizar el handover y que pueden ser diferentes a las proporcionadas en la petición; y donde el primer elemento es el de mayor preferencia.

Tipo de dato estándar IEEE 802.21: LIST(RQ_RESULT)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la PreferredCandidateLinkList.

● ***MIH_N2N_HO_Commit_Request***

Esta primitiva es usada por los usuarios MIH de la Serving Network para informar a la red candidata que ha sido elegida para el handover que va a realizar el MN.

Se encarga de construir el mensaje de petición (MIH_N2N_HO_Commit Request) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Request descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: Identificador de la MIHF remota destino de la petición.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ MNIdentifier: Identificador de la MIHF del MN que va a realizar el handover.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MNIdentifier.

→ TargetMNLinkIdentifier: Identificador del enlace del MN para el que se solicitan recursos.

Tipo de dato estándar IEEE 802.21: LINK_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetMNLinkIdentifier.

→ TargetPoA: Dirección de enlace del PoA (AP/BS) destino del handover.

Tipo de dato estándar IEEE 802.21: LINK_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetPoA.

→ RequestedResourceSet: Conjunto de parámetros requeridos para el control de admisión del MN y la reserva de recursos en la red candidata para el handover.

Tipo de dato estándar IEEE 802.21: REQ_RES_SET
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el RequestedResourceSet.

● ***MIH_MN_HO_Commit_Indication***

Esta primitiva es usada por la MIHF para indicar que se ha recibido el mensaje MIH_MN_HO_Commit Request desde el MN.

Se encarga de realizar la llamada a la primitiva MIH_N2N_HO_Commit Request a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Indication descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: MIHF ID del nodo que ha enviado el mensaje MIH_MN_HO_Commit Request

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ LinkType:Tipo de enlace de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: LINK_TYPE
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ TargetNetworkInfo: Información de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: TGT_NET_INFO
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la TargetNetworkInfo.

● ***MIH_MN_HO_Commit_Response***

Esta primitiva es usada por los usuarios MIH de la Serving Network para comunicarse con los usuarios MIH de la entidad asociada en el MN que envió el mensaje de petición (MIH_MN_HO_Commit_Response).

Se encarga de construir el mensaje MIH_MN_HO_Commit Response (con los datos obtenidos de la red candidata) como respuesta al mensaje de petición del MN. Su estructura C se corresponde con el tipo HO Response descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

DestinationIdentifier: MIHF ID del MN que envió el mensaje MIH_MN_HO_Commit Request

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento

→ LinkType: Tipo de enlace de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: LINK_TYPE
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ TargetNetworkInfo: Información de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: TGT_NET_INFO
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la TargetNetworkInfo.

● ***MIH_N2N_HO_Commit_Confirm***

Esta primitiva es usada por la MIHF de la Serving Network para informar de la recepción de un mensaje MIH_N2N_HO_Commit Response Response enviado desde la red seleccionada para el handover.

Esta primitiva se encarga de llamar a la primitiva MIH_MN_HO_Commit_Response para construir el mensaje de respuesta a la petición del MN. Su estructura C se corresponde con el tipo HO Confirm descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: MIHF ID del nodo que ha enviado el mensaje MIH_N2N_HO_Commit Response

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ MNIdentifier: Identificador de la MIHF del MN que va a realizar el handover.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MNIdentifier.

→ TargetLinkIdentifier: Identificador del PoA (AP/BS) para el MN.

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

→ AssignedResourceSet: Conjunto de parámetros de recursos asignados por la red candidata al MN.

Tipo de dato estándar IEEE 802.21: ASGN_RES_SET

Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el RequestedResourceSet.

● ***MIH_N2N_HO_Complete_Response***

Esta primitiva es usada para enviar la respuesta a una petición de finalización del handover.

Se encarga de construir el mensaje de respuesta (MIH_N2N_HO_Complete Response) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Response descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: MIHF ID del MN que envió el mensaje MIH_N2N_HO_Complete Request

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ MNIdentifier: Identificador de la MIHF del MN que va a realizar el handover.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MNIdentifier.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ TargetLinkIdentifier: Enlace objetivo del handover.

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

→ ResourceRetentionSet: Estado de los recursos locales del nodo que invoca la primitiva.

Tipo de dato estándar IEEE 802.21: LINK_RR_STATUS
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el ResourceRetentionSet.

● ***MIH_N2N_HO_Complete_Indication***

Esta primitiva es usada por la MIHF para indicar que se ha recibido el mensaje MIH_N2N_HO_Complete Request desde la red donde se ha realizado el handover informando del estado del mismo.

Se encarga de realizar la llamada a la primitiva MIH_N2N_HO_Complete Response a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Indication descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: MIHF ID del nodo que ha enviado el mensaje MIH_N2N_HO_Complete Request

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ MNIdentifier: Identificador de la MIHF del MN.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MNIdentifier.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ TargetLinkIdentifier: Enlace objetivo del handover.

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

→ HandoverResult: Resultado del handover

Tipo de dato estándar IEEE 802.21: HO_RESULT
Tipo de dato C: unsigned char. Un byte de almacenamiento.

3.5.1.3 Candidate Networks

A continuación se describen las primitivas usadas por las redes candidatas (Candidate Networks) durante el proceso de handover iniciado por el MN y mostrado en la Figura 3.11.

● ***MIH_N2N_HO_Query_Resources_Response***

Esta primitiva es usada por la MIHF de la red candidata para comunicarse con su entidad asociada en la Serving Network que envió la petición.

Se encarga de construir el mensaje de respuesta (MIH_N2N_HO_Query_Resources Response) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Response descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: MIHF ID del nodo que ha enviado el mensaje MIH_N2N_HO_Query_Resources Request

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ ResourceStatus: Especifica si los recursos están o no disponibles en el nuevo PoA

Tipo de dato estándar IEEE 802.21: LINK_RES_STATUS
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la CandidateLinkList.

→ CandidateLinkList: Lista de elementos RQ_RESULT de la red candidata

Tipo de dato estándar IEEE 802.21: LIST(RQ_RESULT)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la CandidateLinkList.

→ IPAddressInformationStatus: Disponibilidad de los métodos de configuración IP

Tipo de dato estándar IEEE 802.21: IP_CFG_STATUS
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la CandidateLinkList.

● ***MIH_N2N_HO_Query_Resources_Indication***

Esta primitiva indica que se ha recibido el mensaje MIH_N2N_HO_Query_Resources Request desde la Serving Network.

Se encarga de realizar la llamada a la primitiva (MIH_N2N_HO_Query_Resources Response) a partir de los parámetros

proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Indication descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: MIHF ID del nodo que ha enviado el mensaje MIH_N2N_HO_Candidate_Query Request

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ QoSResourceRequirements: QoS para los recursos mínima en la red candidata

Tipo de dato estándar IEEE 802.21: QOS_LIST
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena los QoSResourceRequirements.

→ IPConfigurationMethods (Opcional): Métodos de configuración IP actuales

Tipo de dato estándar IEEE 802.21: IP_CFG_MTHDS
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena los IPConfigurationMethods. Su valor es NULL cuando no es utilizado.

→ DHCPServerAddress (Opcional): Dirección IP del servidor DHCP actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la DHCPServerAddress. Su valor es NULL cuando no es utilizado.

→ FAAdresss (Opcional): Dirección IP del servidor Foreign Agent actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la FAAdress. Su valor es NULL cuando no es utilizado.

→AccessRouterAdresss (Opcional): Dirección IP del router de acceso actual

Tipo de dato estándar IEEE 802.21: IP_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el AccessRouterAdress. Su valor es NULL cuando no es utilizado.

→ CandidateLinkList: Lista de enlaces candidatos (APs o BSs) de una determinada red candidata. Cada enlace se identifica por su tipo y por la dirección de enlace del PoA

Tipo de dato estándar IEEE 802.21: LIST(LINK_ID)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la CandidateLinkList.

● ***MIH_N2N_HO_Commit_Response***

Esta primitiva es usada para construir el mensaje de respuesta a la petición acerca de la asignación de recursos realizada por la Serving Network.

Se encarga de construir el mensaje MIH_N2N_HO_Commit Response como respuesta al mensaje de petición de la Serving Network. Su estructura C se corresponde con el tipo HO Response descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: MIHF ID del MN que envió el mensaje MIH_MN_HO_Commit Request

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ MNIdentifier: Identificador de la MIHF del MN.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MNIdentifier.

→ TargetLinkIdentifier: Identificador del PoA (AP/BS) para el MN.

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

→ AssignedResourceSet: Conjunto de parámetros de recursos asignados por la red candidata al MN.

Tipo de dato estándar IEEE 802.21: ASGN_RES_SET
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el RequestedResourceSet.

● ***MIH_N2N_HO_Commit_Indication***

Esta primitiva es usada por los usuarios MIH para informar de que se ha recibido en un mensaje MIH_N2N_HO_Commit_Request.

Se encarga de realizar la llamada a la primitiva MIH_N2N_HO_Commit Response a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Indication descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: Identifica la MIHF que invoca la primitiva

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ MNIdentifier: Identificador de la MIHF del MN.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MNIdentifier.

→ TargetMNLinkIdentifier: Dirección del enlace del MN para el cual se van a reservar recursos.

Tipo de dato estándar IEEE 802.21: LINK_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

→ TargetPoA: Dirección de enlace del PoA objetivo del handover.

Tipo de dato estándar IEEE 802.21: LINK_ADDR
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetPoA.

→ RequestedResourceSet: Conjunto de parámetros requeridos para el control de admisión del MN y la reserva de recursos en la red candidata para el handover.

Tipo de dato estándar IEEE 802.21: REQ_RES_SET
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el RequestedResourceSet.

● ***MIH_N2N_HO_Complete_Request***

Esta primitiva es usada por la red candidata para informar a la Serving Network sobre la finalización del handover.

Se encarga de construir el mensaje de petición (MIH_N2N_HO_Complete Request) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Request descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: Identifica el MIHF ID del nodo destino del mensaje

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ MNIdentifier: Identificador de la MIHF del MN.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MNIdentifier.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ TargetLinkIdentifier: Enlace objetivo del handover.

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

→ HandoverResult: Resultado del handover

Tipo de dato estándar IEEE 802.21: HO_RESULT
Tipo de dato C: unsigned char. Un byte de almacenamiento.

● ***MIH_MN_HO_Complete_Indication***

Esta primitiva es usada para informar de que se ha recibido un mensaje MIH_MN_HO_Complete_Request.

Se encarga de realizar la llamada a la primitiva MIH_N2N_HO_Complete Request a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Indication descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: Identifica el MIHF ID del nodo que envió el mensaje MIH_MN_HO_Complete Request

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ MNIdentifier: Identificador de la MIHF del MN.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MNIdentifier.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ TargetLinkIdentifier: Enlace objetivo del handover.

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

→ HandoverResult: Resultado del handover

Tipo de dato estándar IEEE 802.21: HO_RESULT
Tipo de dato C: unsigned char. Un byte de almacenamiento.

● ***MIH_MN_HO_Complete_Response***

Esta primitiva es usada por los usuarios MIH para enviar una respuesta a la petición del MN.

Se encarga de construir el mensaje de respuesta (MIH_MN_HO_Complete Response) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Response descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: MIHF ID del MN que envió el mensaje MIH_MN_HO_Complete Request

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ TargetLinkIdentifier: Enlace objetivo del handover.

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

● ***MIH_N2N_HO_Complete_Confirm***

Esta primitiva es usada por la MIHF para informar de la recepción de un mensaje MIH_N2N_HO_Complete Response desde una entidad MIHF asociada.

Esta primitiva se encarga de llamar a la primitiva MIH_MN_HO_Complete_Response para construir el mensaje de respuesta a la petición del MN. Su estructura C se corresponde con el tipo HO Confirm descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→SourceIdentifier: Identifica el MIHF ID del nodo que envió el mensaje MIH_N2N_HO_Complete Response

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ MNIdentifier: Identificador de la MIHF del MN.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el MNIdentifier.

→ SourceLinkIdentifier: Enlace fuente del handover

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ TargetLinkIdentifier: Enlace objetivo del handover.

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el TargetLinkIdentifier.

→ ResourceRetentionStatus: Estado de los recursos locales del nodo que invoca la primitiva MIH_N2N_HO_Complete Response.

Tipo de dato estándar IEEE 802.21: LINK_RR_STATUS
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el ResourceRetentionSet.

3.5.2 Mensajes

A continuación se describen los mensajes usados por el MN, la Serving Network y las Redes Candidatas durante el proceso de handover iniciado por el MN y mostrado en la Figura 3.11.

Además de la cabecera, los campos Source Identifier y Destination Identifier se incluyen en todos los mensajes:

→ Source Identifier (Source MIHF ID TLV):

Valor del campo tipo de la estructura TLV: 1
Tipo de dato estándar IEEE 802.21: MIHF_ID

→ Destination Identifier (Destination MIHF ID TLV):

Valor del campo tipo de la estructura TLV: 2
Tipo de dato estándar IEEE 802.21: MIHF_ID

● ***MIH_MN_HO_Candidate_Query Request***

Usado por la MIHF del MN para comunicar a la MIHF de la red un intento de iniciar el handover.

Cabecera: SID = 3, Opcode = 1, AID = 5

CAMPOS:

→ SourceLinkIdentifier (Link identifier TLV):

Valor del campo tipo de la estructura TLV: 13
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ CandidateLinkList (List of link PoA list TLV):

Valor del campo tipo de la estructura TLV: 31
Tipo de dato estándar IEEE 802.21: LIST(LINK_POA_LIST)

→ QoSResourceRequirements (Handover resource query list TLV):

Valor del campo tipo de la estructura TLV: 33
Tipo de dato estándar IEEE 802.21: QOS_LIST

→ IPConfigurationMethods (IP address configuration methods TLV) (Opcional):

Valor del campo tipo de la estructura TLV: 19
Tipo de dato estándar IEEE 802.21: IP_CFG_MTHDS

→ DHCPServerAddress (DHCP Server address TLV) (Opcional):

Valor del campo tipo de la estructura TLV: 36
Tipo de dato estándar IEEE 802.21: IP_ADDR

→ FAAdresss (FA address TLV) (Opcional):

Valor del campo tipo de la estructura TLV: 37
Tipo de dato estándar IEEE 802.21: IP_ADDR

→ AccessRouterAdresss (Access router address TLV) (Opcional):

Valor del campo tipo de la estructura TLV: 35
Tipo de dato estándar IEEE 802.21: IP_ADDR

● ***MIH_N2N_HO_Query_Resources Request***

Usado por la MIHF de la Serving Network para comunicar a la MIHF de la red candidata un intento de handover. También se utiliza para obtener información relativa al direccionamiento IP de la red candidata.

Cabecera: SID = 3, Opcode = 1, AID = 6

CAMPOS:

→ QoSResourceRequirements (Handover resource query list TLV):

Valor del campo tipo de la estructura TLV: 33
Tipo de dato estándar IEEE 802.21: QOS_LIST

→ IPConfigurationMethods (IP address configuration methods TLV) (Opcional):

Valor del campo tipo de la estructura TLV: 19
Tipo de dato estándar IEEE 802.21: IP_CFG_MTHDS

→ DHCPServerAddress (DHCP Server address TLV) (Opcional):

Valor del campo tipo de la estructura TLV: 36
Tipo de dato estándar IEEE 802.21: IP_ADDR

→ FAAdresss (FA address TLV) (Opcional):

Valor del campo tipo de la estructura TLV: 37
Tipo de dato estándar IEEE 802.21: IP_ADDR

→ AccessRouterAdresss (Access router address TLV) (Opcional):

Valor del campo tipo de la estructura TLV: 35
Tipo de dato estándar IEEE 802.21: IP_ADDR

● ***MIH_N2N_HO_Query_Resources Response***

Usado por la de la red candidata como respuesta al mensaje de petición MIH_N2N_HO_Query_Resources Request de la Serving Network. El mensaje contiene el resultado de la preparación de los recursos y el estado del enlace para el handover e información relativa al direccionamiento IP.

Cabecera: SID = 3, Opcode = 2, AID = 6

CAMPOS:

→ Status (Status TLV):

Valor del campo tipo de la estructura TLV: 3
Tipo de dato estándar IEEE 802.21: STATUS

→ ResourceStatus (Resource status TLV) (No incluído si el estado no es Success):

Valor del campo tipo de la estructura TLV: 42
Tipo de dato estándar IEEE 802.21: LINK_RES_STATUS

→ CandidateLink List (Preferred link list TLV) (No incluído si el estado no es Success) (Opcional):

Valor del campo tipo de la estructura TLV: 32
Tipo de dato estándar IEEE 802.21: LIST(RQ_RESULT)

→ IPAddressInformationStatus (IP address information status TLV) No incluído si el estado no es Success) (Opcional):

Valor del campo tipo de la estructura TLV: 38
Tipo de dato estándar IEEE 802.21: IP_CFG_STATUS

● ***MIH_MN_HO_Candidate_Query Response***

Usado por la MIHF de la Serving Network como respuesta al mensaje de petición MIH_MN_HO_Candidate_Query Request del MN.

Cabecera: SID = 3, Opcode = 2, AID = 5

CAMPOS:

→ Status (Status TLV):

Valor del campo tipo de la estructura TLV: 3
Tipo de dato estándar IEEE 802.21: STATUS

→ SourceLinkIdentifier (Link identifier TLV):

Valor del campo tipo de la estructura TLV: 13

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ PreferredCandidateLink List (Preferred link list TLV) (No incluído si el estado no es Success):

Valor del campo tipo de la estructura TLV: 32
Tipo de dato estándar IEEE 802.21: LIST(RQ_RESULT)

● MIH_MN_HO_Commit Request

Usado por la MIHF del MN para comunicar al PoS de la Serving Network la información de la red elegida para el handover.

Cabecera: SID = 3, Opcode = 1, AID = 7

CAMPOS:

→ Link Type (Link type TLV):

Valor del campo tipo de la estructura TLV: 4
Tipo de dato estándar IEEE 802.21: LINK_TYPE

→ TargetNetworkInfo (Target network info TLV):

Valor del campo tipo de la estructura TLV: 56
Tipo de dato estándar IEEE 802.21: TGT_NET_INFO

● ***MIH_N2N_HO_Commit Request***

Usado por la MIHF de la Serving Network para comunicar con su par MIHF en la red seleccionada. Se utiliza para solicitar a la red candidata la asignación de recursos para el MN que va a realizar el handover

Cabecera: SID = 3, Opcode = 1, AID = 9

CAMPOS:

→ MNIdentifier (Mobile node MIHF ID TLV):

Valor del campo tipo de la estructura TLV: 53
Tipo de dato estándar IEEE 802.21: MIHF_ID

→ TargetMobileNodeLinkIdentifier (MN link ID TLV):

Valor del campo tipo de la estructura TLV: 60
Tipo de dato estándar IEEE 802.21: LINK_ID

→ TagetPoA (PoA TLV):

Valor del campo tipo de la estructura TLV: 61

Tipo de dato estándar IEEE 802.21: LINK_ADDR

→ RequestedResourceSet (Requested resource set TLV):

Valor del campo tipo de la estructura TLV: 64
Tipo de dato estándar IEEE 802.21: REQ_RES_SET

● ***MIH_N2N_HO_Commit Response***

Usado por la MIHF de la red candidata seleccionada para comunicar con su par MIHF en la Serving Network. Se utiliza para responder al mensaje MIH_N2N_HO_Commit Request

Cabecera: SID = 3, Opcode = 2, AID = 9

CAMPOS:

→ Status (Status TLV):

Valor del campo tipo de la estructura TLV: 3
Tipo de dato estándar IEEE 802.21: STATUS

→ MNIdentifier (Mobile node MIHF ID TLV):

Valor del campo tipo de la estructura TLV: 53
Tipo de dato estándar IEEE 802.21: MIHF_ID

→ TargetLinkIdentifier (Link identifier TLV) (No incluído si el estado no es Success):

Valor del campo tipo de la estructura TLV: 13
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ AssignedResourceSet (Assigned resource set TLV) (No incluído si el estado no es Success):

Valor del campo tipo de la estructura TLV: 58
Tipo de dato estándar IEEE 802.21: ASGN_RES_SET

● ***MIH_MN_HO_Commit Response***

Usado por la MIHF de la PoS de la Serving Network para responder al mensaje MIH_MN_HO_Commit Request

Cabecera: SID = 3, Opcode = 2, AID = 7

CAMPOS:

→ Status (Status TLV):

Valor del campo tipo de la estructura TLV: 3

Tipo de dato estándar IEEE 802.21: STATUS

→ Link Type (Link type TLV):

Valor del campo tipo de la estructura TLV: 4
Tipo de dato estándar IEEE 802.21: LINK_TYPE

→ TargetNetworkInfo (Target network info TLV):

Valor del campo tipo de la estructura TLV: 56
Tipo de dato estándar IEEE 802.21: TGT_NET_INFO

● ***MIH_MN_HO_Complete Request***

Usado por la MIHF del MN para comunicar el estado del handover a la MIHF de la red seleccionada.

Cabecera: SID = 3, Opcode = 1, AID = 10

CAMPOS:

→ SourceLinkIdentifier (Link identifier TLV):

Valor del campo tipo de la estructura TLV: 13
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ TargetLinkIdentifier (New link identifier TLV):

Valor del campo tipo de la estructura TLV: 14
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ HandoverResult (Handover result TLV):

Valor del campo tipo de la estructura TLV: 41
Tipo de dato estándar IEEE 802.21: HO_RESULT

● ***MIH_N2N_HO_Complete Request***

Usado por la MIHF para comunicar el estado del handover

Cabecera: SID = 3, Opcode = 1, AID = 11

CAMPOS:

→ MNIdentifier (Mobile node MIHF ID TLV):

Valor del campo tipo de la estructura TLV: 53
Tipo de dato estándar IEEE 802.21: MIHF_ID

→ SourceLinkIdentifier (Link identifier TLV):

Valor del campo tipo de la estructura TLV: 13
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ TargetLinkIdentifier (New link identifier TLV):

Valor del campo tipo de la estructura TLV: 14
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ HandoverResult (Handover result TLV):

Valor del campo tipo de la estructura TLV: 41
Tipo de dato estándar IEEE 802.21: HO_RESULT

● ***MIH_N2N_HO_Complete Response***

Usado por la MIHF para comunicar que se ha completado el handover.

Cabecera: SID = 3, Opcode = 2, AID = 11

CAMPOS:

→ Status (Status TLV):

Valor del campo tipo de la estructura TLV: 3
Tipo de dato estándar IEEE 802.21: STATUS

→ MNIdentifier (Mobile node MIHF ID TLV):

Valor del campo tipo de la estructura TLV: 53
Tipo de dato estándar IEEE 802.21: MIHF_ID

→ SourceLinkIdentifier (Link identifier TLV):

Valor del campo tipo de la estructura TLV: 13
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ TargetLinkIdentifier (New link identifier TLV):

Valor del campo tipo de la estructura TLV: 14
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ ResourceRetentionStatus (Resource retention status TLV) (No incluído si el estado no es Success):

Valor del campo tipo de la estructura TLV: 43
Tipo de dato estándar IEEE 802.21: LINK_RR_STATUS

● ***MIH_MN_HO_Complete Response***

Usado por la MIHF de la red seleccionada para comunicar al MN que se ha completado el handover

Cabecera: SID = 3, Opcode = 2, AID = 10

CAMPOS:

→ Status (Status TLV):

Valor del campo tipo de la estructura TLV: 3
Tipo de dato estándar IEEE 802.21: STATUS

→ SourceLinkIdentifier (Link identifier TLV):

Valor del campo tipo de la estructura TLV: 13
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ TargetLinkIdentifier (New link identifier TLV):

Valor del campo tipo de la estructura TLV: 14
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

3.6 Handover iniciado por la Red

El procedimiento para el handover iniciado por la Red se muestra en la siguiente figura (Figura 3.12):

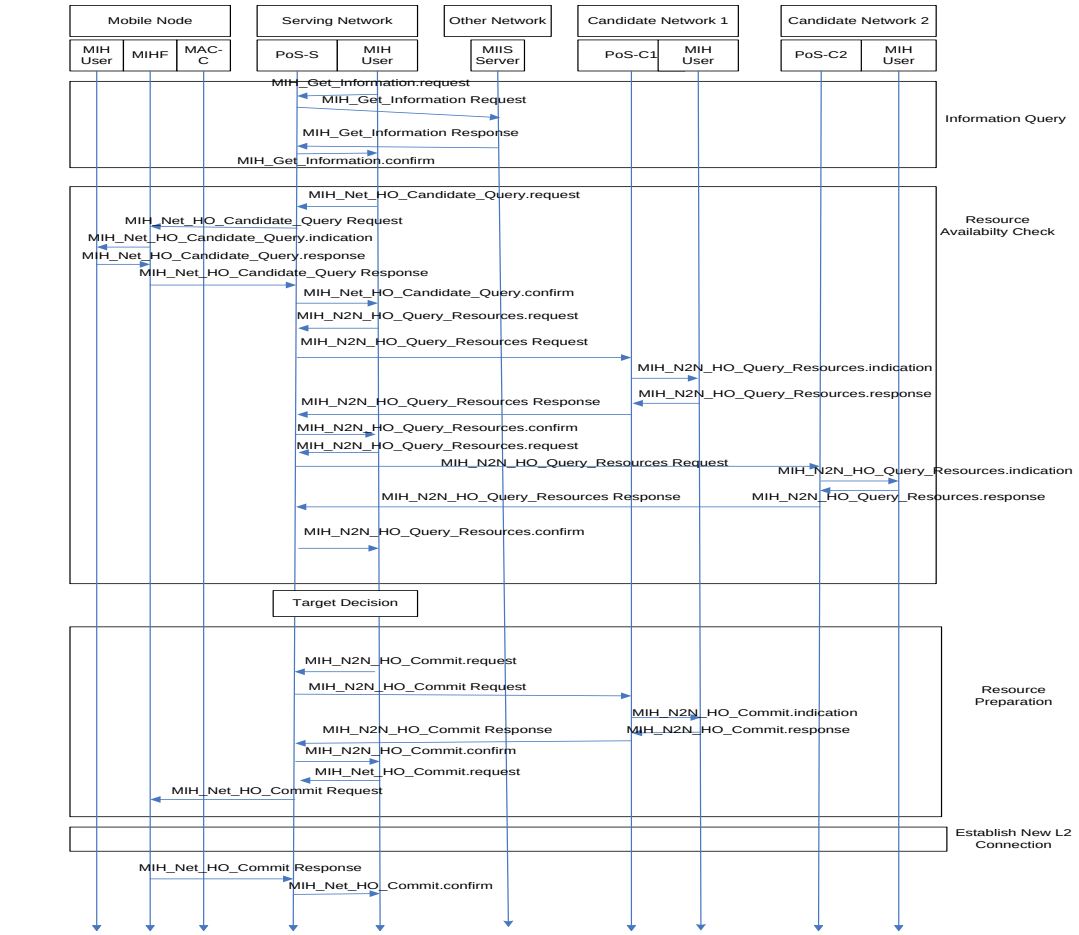

Figura 3.12: Handover iniciado por la Red

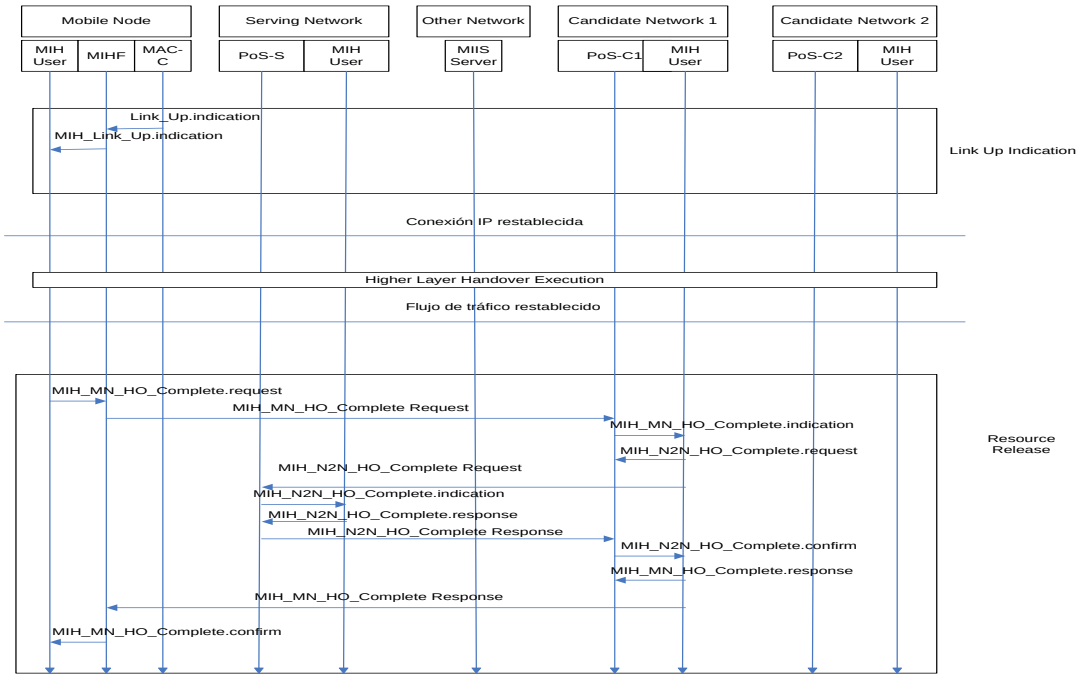

Figura 3.12 (Continuación): Handover iniciado por la Red

La Serving Network solicita información al MIH Information Server (MIIS) acerca de sus redes vecinas mediante el mensaje MIH_Get_Information Request. El MIIS responde mediante el mensaje MIH_Get_Information Response con la información disponible.

La Serving Network empieza la primera fase del handover (Resource Availability Check) mediante el envío del mensaje MIH_Net_HO_Candidate Query Request al MN que responde mediante el mensaje MIH_Net_HO_Candidate Query Response que contiene el asentimiento del MN y la lista de las redes para el handover. La Serving Network solicita dicha información a cada una de las redes candidatas mediante el envío del mensaje MIH_N2N_HO_Query _Resorces Request. Cada una de estas redes candidatas responde con la información solicitada mediante el mensaje MIH_N2N_HO_Query _Resorces Response.

La segunda fase del handover (Resource Preparation) comienza con el envío, por parte de la Serving Network, del mensaje MIH_N2N_HO_Commit Request a la red seleccionada preparando los recursos para realizar el handover. Cuando la Serving Network tiene información de dichos recursos envía el mensaje de petición MIH_Net_HO_Commit Request al MN para la asignación de recursos del handover.

Cuando se ha establecido la conexión a nivel 2 el MN envía el mensaje MIH_N2N_HO_Commit Response a la Serving Network**.**

Una vez que se ha reestablecido la conectividad IP y el flujo de datos de las capas superiores, se inicia la última fase del handover (Resource Release) . Esta fase se inicia con el envío del mensaje MIH_MN_HO_Complete Request por parte del MN a la red donde se ha realizado el handover para informar del resultado del mismo. La red candidata informa a la antigua Serving Network del resultado del handover y le envía una petición mediante el mensaje MIH_N2N_HO_Complete Request para la liberación de los recursos que estaba usando el MN en dicha red. La información del estado de liberación de los recursos es enviada al MN mediante el mensaje MIH_MN_HO_Complete Response por la red candidata, que ha pasado ha ser la nueva Serving Network.

3.6.1 Primitivas

3.6.1.1 Mobile Node

A continuación se describen las primitivas usadas por el MN durante el proceso de handover iniciado por la Red y mostrado en la Figura 3.12.

● ***MIH_Net_HO_Candidate Query Indication***

Esta primitiva es usada la MIHF para indicar a un usuario MIH que se ha recibido el mensaje MIH_Net_HO_Candidate_Query Request desde una MIHF remota.

Se encarga de realizar la llamada a la primitiva MIH_Net_HO_Candidate_Query_Response para mandar información a la Serving Network. Su estructura C se corresponde con el tipo HO Indication descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→SourceIdentifier: MIHF remota que es la que invoca la primitiva

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ SuggestedNewLinkList:Lista de PoAs (para cada enlace) candidatos para realizar el handover, donde el primer elemento es el de mayor preferencia.

Tipo de dato estándar IEEE 802.21: LIST(LINK_POA_LIST)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SuggestedNewLinkList.

→ QueryResourceReportFlag: Especifica si el MN tiene que reportar los recursos.

Tipo de dato estándar IEEE 802.21: BOOLEAN
Tipo de dato C: unsigned char. Un byte de almacenamiento.

● ***MIH_Net_HO_Candidate Query Response***

Esta primitiva es usada la MIHF del MN como respuesta a la petición de la Serving Network.

Se encarga de construir el mensaje MIH_Net_HO_Candidate_Query Response como respuesta al mensaje de petición de la Serving Network. Su estructura C se corresponde con el tipo HO Response descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: Identificador de la MIHF remota destino de la respuesta

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ SourceLinkIdentifier: Enlace actual

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ HandoverStatus: Estado de la petición del handover

Tipo de dato estándar IEEE 802.21: HO_STATUS
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena los IPConfigurationMethods. Su valor es NULL cuando no es utilizado.

→ PreferredLinkList:Lista cuyos elementos son datos son del tipo RQ_RESULT que contienen las redes que deberían considerarse para realizar el handover y que pueden ser diferentes a las proporcionadas en la petición; y donde el primer elemento es el de mayor preferencia.

Tipo de dato estándar IEEE 802.21: LIST(RQ_RESULT)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la PreferredLinkList.

3.6.1.2 Serving Network

A continuación se describen las primitivas usadas por la Serving Network durante el proceso de handover iniciado por la Red y mostrado en la Figura 3.12.

● ***MIH_Net_HO_Candidate Query Request***

Esta primitiva es usada por los usuarios MIH de la red para informar a los usuarios MIH de su entidad par sobre la intención de realizar un handover.

Se encarga de construir el mensaje de petición (MIH_Net_HO_Candidate_Query Request) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Request descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: Identificador de la MIHF remota que es el destinatario del mensaje MIH

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ SuggestedNewLinkList: Lista de PoAs (para cada enlace) candidatos para realizar el handover, donde el primer elemento es el de mayor preferencia.

Tipo de dato estándar IEEE 802.21: LIST(LINK_POA_LIST)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SuggestedNewLinkList.

→ QueryResourceReportFlag: Especifica si el MN tiene que reportar los recursos.

Tipo de dato estándar IEEE 802.21: BOOLEAN
Tipo de dato C: unsigned char. Un byte de almacenamiento.

● ***MIH_Net_HO_Candidate Query Confirm***

Esta primitiva es usad por la MIHF para confirmar que se ha recibido el mensaje MIH_Net_HO_Candidate Query Response de su entidad MIHF par.

Esta primitiva se encarga de llamar a la primitiva MIH_N2N_HO_Query_Resources_Request para construir el mensaje de petición a la red candidata. Su estructura C se corresponde con el tipo HO Confirm descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: MIHF ID del MN que envió el mensaje MIH_Net_HO_Candidate Query Response

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ SourceLinkIdentifier: Enlace actual

Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceLinkIdentifier.

→ HandoverStatus: Estado de la petición del handover

Tipo de dato estándar IEEE 802.21: HO_STATUS

Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena los IPConfigurationMethods. Su valor es NULL cuando no es utilizado.

→ PreferredLinkList: Lista cuyos elementos son datos son del tipo RQ_RESULT que contienen las redes que deberían considerarse para realizar el handover y que pueden ser diferentes a las proporcionadas en la petición; y donde el primer elemento es el de mayor preferencia.

Tipo de dato estándar IEEE 802.21: LIST(RQ_RESULT)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la PreferredLinkList.

● ***MIH_Net_HO_Commit Request***

Esta primitiva es usada por los usuarios MIH de la red para comunicarse con el usurario MIH remoto del MN.

Se encarga de construir el mensaje de petición (MIH_Net_HO_Commit Request) a partir de los parámetros proporcionados a la primitiva. Su estructura C se corresponde con el tipo HO Request descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ DestinationIdentifier: MIHF ID de la MIHF del Mobile Node que realiza el handover.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el DestinationIdentifier.

→ LinkType: Tipo de enlace de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: LINK_TYPE
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ TargetNetworkInfoList: Información de la red elegida para el handover que ayuda al MN en el proceso.

Tipo de dato estándar IEEE 802.21: LIST(TGT_NET_INFO)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la TargetNetworkInfoList.

→ AssignedResourceSet: Conjunto de parámetros de recursos asignados por la red candidata al MN para el handover.

Tipo de dato estándar IEEE 802.21: ASGN_RES_SET
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el RequestedResourceSet.

→ LinkActionExecutionDelay: Tiempo, en ms, que se ha de esperar antes de ejecutar la acción.

Tipo de dato estándar IEEE 802.21: UNSIGNED_INT(2)
Tipo de dato C: unsigned short. Dos bytes de almacenamiento.

→ LinkActionList (Opcional): Lista de acciones a realizar sobre el enlace.

Tipo de dato estándar IEEE 802.21: LIST(LINK_ACTION_REQ)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la LinkActionList.

● ***MIH_Net_HO_Commit Confirm***

Esta primitiva es usada por la MIHF de la Serving Network para confirmar que se ha recibido el mensaje MIH_Net_HO_Commit Response enviado por la entidad MIHF del MN.

Se encarga de modificar el valor de la variable Resource Preparation, cambiando su valor a TRUE, para indicar que se ha llegado al final de la fase Resource Preparation del handover. Su estructura C se corresponde con el tipo HO Confirm descrito en el apartado 3.3.

PARÁMETROS DE ENTRADA:

→ SourceIdentifier: MIHF ID del MN que envió el mensaje MIH_MN_HO_Commit Response.

Tipo de dato estándar IEEE 802.21: MIHF_ID
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena el SourceIdentifier.

→ Status: Estado de la operación

Tipo de dato estándar IEEE 802.21: STATUS
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ LinkType: Tipo de enlace de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: LINK_TYPE
Tipo de dato C: unsigned char. Un byte de almacenamiento.

→ TargetNetworkInfo: Información de la red elegida para el handover

Tipo de dato estándar IEEE 802.21: TGT_NET_INFO
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la TargetNetworkInfo.

→ LinkActionResultList (Opcional): Lista con el resultado de las acciones realizadas sobre el enlace.

Tipo de dato estándar IEEE 802.21: LIST(LINK_ACTION_RSP)
Tipo de dato C: char *. Puntero al inicio de la secuencia de bytes que almacena la LinkActionResultList.

3.6.1.3 Candidate Networks

Las primitivas usadas por las redes candidatas durante el proceso de handover iniciado por la Red y mostrado en la Figura 3.12 son las mismas que las descritas en el apartado 3.5.1.3.

3.6.2 Mensajes

A continuación se describen los mensajes usados por el MN, la Serving Network y las Redes Candidatas durante el proceso de handover iniciado por la Red y mostrado en la Figura 3.12.

Además de la cabecera, los campos Source Identifier y Destination Identifier se incluyen en todos los mensajes:

→ Source Identifier (Source MIHF ID TLV):

Valor del campo tipo de la estructura TLV: 1
Tipo de dato estándar IEEE 802.21: MIHF_ID

→ Destination Identifier (Destination MIHF ID TLV):

Valor del campo tipo de la estructura TLV: 2
Tipo de dato estándar IEEE 802.21: MIHF_ID

- ***MIH_MN_HO_Candidate_Query Request***

Usado para la comunicación entre la MIHF de un MN y la MIHF de la red. Comunica un intento de iniciar un handover por parte de la Serving Network.

Cabecera: SID = 3, Opcode = 1, AID = 4

CAMPOS:

→ SuggestedNewLinkList (List of link PoA list TLV):

Valor del campo tipo de la estructura TLV: 31
Tipo de dato estándar IEEE 802.21: LIST(LINK_POA_LIST)

→ QueryResourceReportFlag (Query resource report flag TLV):

Valor del campo tipo de la estructura TLV: 54
Tipo de dato estándar IEEE 802.21: BOOLEAN

● ***MIH_MN_HO_Candidate_Query Response***

Usado para la comunicación entre la MIHF de un MN y la MIHF de la red. Usado para responder a un intento de iniciar un handover por parte de la Serving Network.

Cabecera: SID = 3, Opcode = 2, AID = 4

CAMPOS:

→ Status (Status TLV):

Valor del campo tipo de la estructura TLV: 3
Tipo de dato estándar IEEE 802.21: STATUS

→ SourceLinkIdentifier (Link identifier TLV):

Valor del campo tipo de la estructura TLV: 13
Tipo de dato estándar IEEE 802.21: LINK_TUPLE_ID

→ HandoverStatus (Handover status TLV):

Valor del campo tipo de la estructura TLV: 34
Tipo de dato estándar IEEE 802.21: HO_STATUS

→ PreferredCandidateLink List (Preferred link list TLV) (No incluído si el estado no es Success):

Valor del campo tipo de la estructura TLV: 32
Tipo de dato estándar IEEE 802.21: LIST(RQ_RESULT)

● ***MIH_Net_HO_Commit Request***

Usado por la MIHF para comunicar la intención de asignar un enlace y un PoA específico a una petición realizada durante el handover.

Cabecera: SID = 3, Opcode = 1, AID = 8

CAMPOS:

→ LinkType (Link type TLV) :

Valor del campo tipo de la estructura TLV: 4
Tipo de dato estándar IEEE 802.21: LINK_TYPE

→ TargetNetworkInfoList (List of target network info TLV):

Valor del campo tipo de la estructura TLV: 57
Tipo de dato estándar IEEE 802.21: LIST(TGT_NET_INFO)

→ Assigned ResourceSet (Assigned resource set TLV):

Valor del campo tipo de la estructura TLV: 58
Tipo de dato estándar IEEE 802.21: ASGN_RES_SET

→ LinkActionExecutionDelay (Time interval TLV):

Valor del campo tipo de la estructura TLV: 21
Tipo de dato estándar IEEE 802.21: UNSIGNED_INT(2)

→ LinkActionsList (Link actions list TLV):

Valor del campo tipo de la estructura TLV: 39
Tipo de dato estándar IEEE 802.21: LIST(LINK_ACTION_REQ)

● ***MIH_Net_HO_Commit Response***

Usado por la MIHF para responder a la petición de asignación de un enlace y un PoA específico del handover.

Cabecera: SID = 3, Opcode = 2, AID = 8

CAMPOS:

→ LinkType (Link type TLV) (No incluído si el estado no es Success):

Valor del campo tipo de la estructura TLV: 4
Tipo de dato estándar IEEE 802.21: LINK_TYPE

→ TargetNetworkInfo (Target network info TLV) (No incluído si el estado no es Success):

Valor del campo tipo de la estructura TLV: 56
Tipo de dato estándar IEEE 802.21: TGT_NET_INFO

→ LinkActionsResultList (Link actions result list TLV) (No incluído si el estado no es Success):

Valor del campo tipo de la estructura TLV: 40
Tipo de dato estándar IEEE 802.21: LIST(LINK_ACTION_RSP)

4. Mobile IPv6

Mobile IPv6 [11] es un protocolo que permite a un nodo mantener sus comunicaciones y estar accesible cuando se mueve a través de una red IPv6 [12]. Sin este soporte los paquetes destinados a un nodo que se encuentra fuera de su red local se perderán ya que dicho nodo no estará accesible; además las conexiones a nivel de transporte no podrán ser mantenidas cuando se produce un movimiento a otra red ya que es necesario cambiar la dirección IP del nodo, de tal forma que las conexiones a nivel de transporte deberán establecerse de nuevo.

Con Mobile IPv6 un mobile node es capaz de moverse entre distintos puntos de acceso sin la necesidad de cambiar su home address (dirección permanente asignada a un MN). Los paquetes se enrutan a la home address del MN sin tener en cuenta donde se encuentra éste. De esta forma el nodo continúa estando accesible cuando se mueve a un nuevo enlace, y el movimiento se hace de forma transparente para los protocolos de transporte y niveles superiores y para las aplicaciones.

Mobile IPv6 presenta mejoras con respecto a Mobile IPv4 [13], algunas de estas mejoras son:

- No es necesaria la utilización de Foreign Agents ya que Mobile IPv6 funciona sin la necesidad de añadir ninguna especificación especial en los routers locales.

- El soporte para la optimización de rutas es una parte fundamental del protocolo.

- La optimización de rutas puede operar sin asociaciones de seguridad pre-establecidas.

- Permite que coexistan de manera eficiente la optimización de rutas y el ingress filtering realizado por los routers.

- El servicio Neighbor Unreachability Detection asegura el MN y el router local estén accesibles de manera simétrica.

- Reduce la cantidad de datos que han de ser enviados ya que cuando el MN está fuera de su red local la mayoría paquetes que le son enviados usan una cabecera para el enrutado más que encapsulación IP.

- Usa IPv6 Neighbor Discovery en vez de ARP [14], lo que proporciona mayor robustez al protocolo.

- El mecanismo Dynamic Home Agent Address Discovery devuelve una sola respuesta al MN.

El funcionamiento de Mobile IPv6 es el siguiente:

El MN siempre debe estar accesible a través de su home address, que es una dirección IP asignada al MN y que se forma a partir del prefijo de subred a la que pertenece.

Cuando el MN está en su red local los paquetes se enrutan hacia el nodo mediante el mecanismo habitual.

Cuando el MN se mueve a otra red adquiere un conjunto de direcciones (care-of-addresses), que se forman a partir del prefijo de subred de la red en la que encuentra, que hacen que el nodo sea accesible mediante dicho conjunto de direcciones. Los paquetes dirigidos a dichas direcciones serán, entonces, enrutados al MN. La asociación entre la home address y las care-of-addresses se denomina binding.

El MN, cuando se encuentra fuera de su red local, tiene que registrar sus care-of-addresses con un router que se encuentre en su red local, que recibe el nombre de Home Agent. Para registrarse el MN envía un mensaje denominado Binding Update a su Home Agent, el cual responde con un mensaje denominado Binding Acknowledgement.

Hay dos posibles formas de comunicación entre un MN y un Correspondent Node (CN, Cualquier nodo que establece una comunicación con el MN) [15]:

1.- Mediante un túnel bidireccional (Figura 4.1):

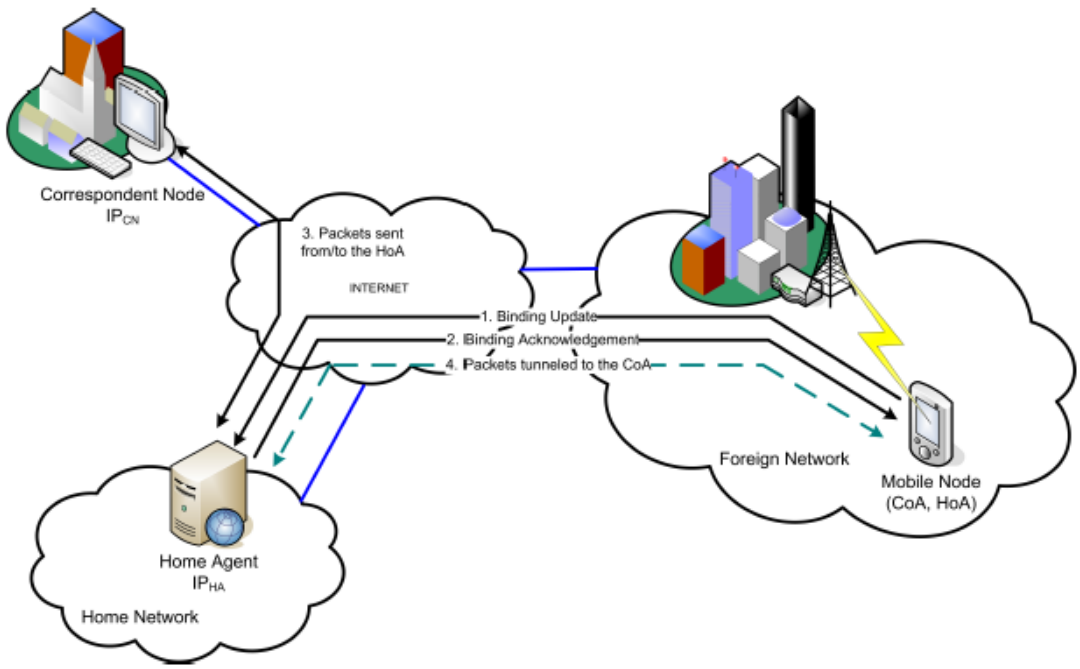

Figura 4.1: Túnel Bidireccional en Mobile IPv6

Los paquetes que tienen como origen el CN son enrutados hacia el Home Agent y posteriormente se reenvían al MN a través de un túnel. Para realizarlo es necesario que el Home Agent intercepte los paquetes dirigidos a la home address del MN en la red local usando para ello el mecanismo Proxy Neighbor Discovery. Cada uno de los paquetes interceptados es enviado a las care-of-addresses del MN a través de un túnel usando la encapsulación IPv6. Los paquetes originados en el MN son enviados al Home

Agent a través del túnel para luego ser enrutados, mediante el mecanismo habitual, hacia el CN.

2.- Mediante el mecanismo de optimización de ruta (Route Optimization): Mediante este mecanismo los paquetes intercambiados entre el MN y el CN se envían directamente de uno a otro y no necesitan pasar por el Home Agent. Para ello es necesario que el MN registre sus care-of-addresses con el CN. De esta forma se reduce el retardo de los mensajes, y elimina la congestión en el Home Agent y en la red local.

Cuando el CN envía los paquetes directamente al MN, el CN introduce la care-of-address del MN en el campo de la dirección de destino de la cabecera del paquete IPv6. Para la Home Address se introduce un nuevo tipo de cabecera de enrutamiento de IPv6. El MN introduce su care-of-address en el campo de la dirección fuente de la cabecera del paquete IPv6 y añade un nuevo campo al paquete IPv6 para llevar su Home Address. Mediante el uso de los campos que llevan la Home Address se consigue que el uso de las care-of-addresses sea transparente para el nivel de transporte.

5. Escenario de pruebas

5.1 Descripción general

Para la realización de las pruebas de funcionamiento del prototipo IEEE 802.21 desarrollado en este proyecto se implementado una red en el laboratorio (Figura 5.1) que consta de los siguientes elementos:

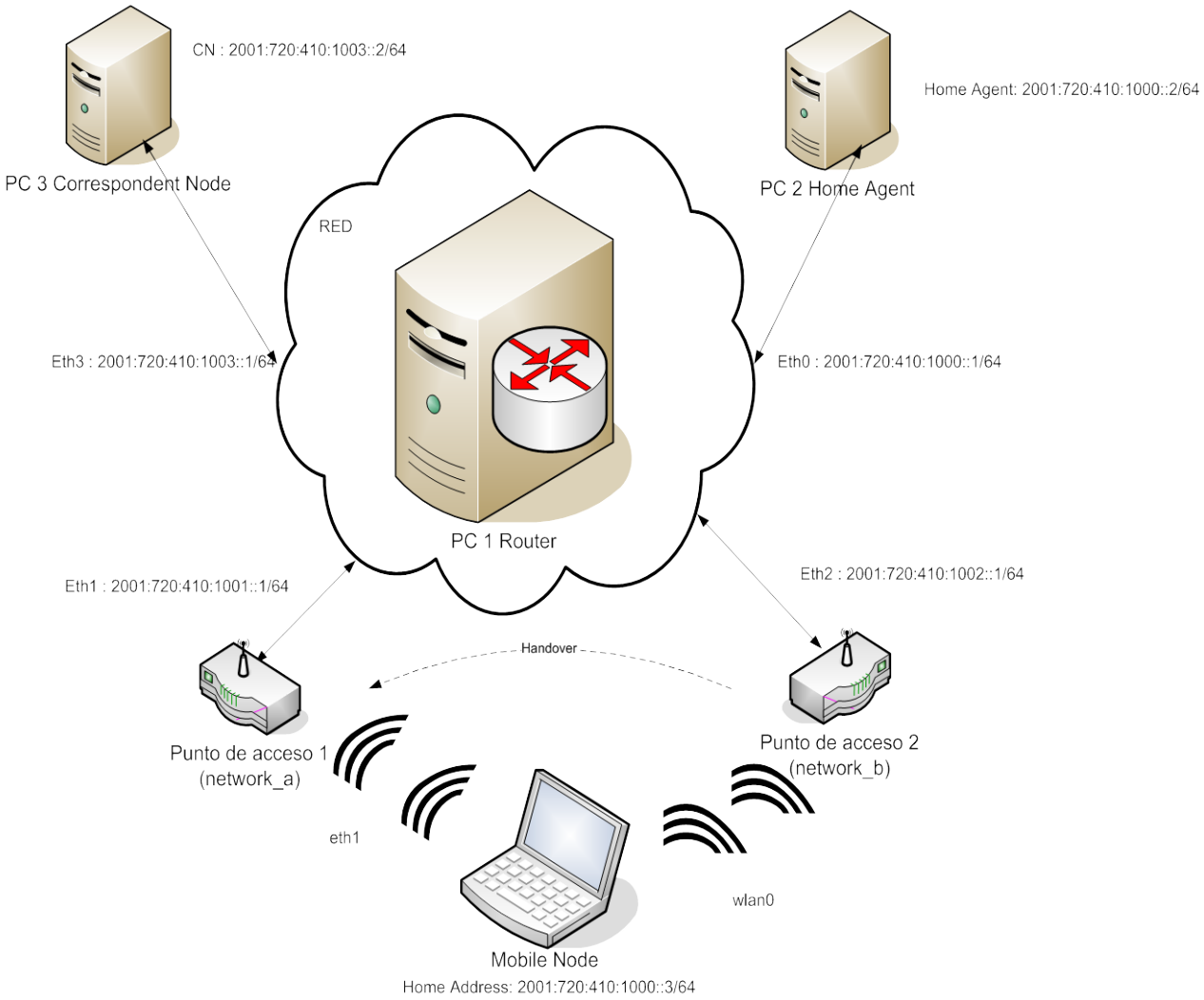

Figura 5.1: Red implementada en el laboratorio

- 3 PCs de sobremesa, cuyas funcionalidades son:

  ● PC 1: Router. Realiza la función de enrutado de los paquetes de la red permitiendo la simulación de Internet.

  ● PC 2: Home Agent. Realiza la función de Home Agent para el MN.

● PC 3: Correspondent Node. Este nodo se utiliza para el intercambio de paquetes con el MN.

- Un ordenador portátil, cuya función es la de MN, que posee de dos tarjetas de red inalámbricas de tipo IEEE 802.11.

- Dos puntos de acceso WiFi (Linksys) [16], que se usan como puntos de acceso de redes diferentes entre las cuales se va a realizar el handover.

En cada uno de los 3PCs y en el ordenador portátil se han instalado los siguientes componentes:

- Distribución Linux Ubuntu 9.04 [17].

- Versión 2.6.31 del kernel de Linux [18], que es necesario compilar con los parámetros de soporte de Mobile IPv6 [19] [20](Anexo 1).

- Parche USAGI Mobile IPv6 [21] para Linux, que implementa las funcionalidades del protocolo Mobile IPv6 (Anexo 2).

- Analizador de protocolos Wireshark [22], para el análisis de los mensajes intercambiados entre los distintos nodos de la red. Para la decodificación de los nuevos mensajes IEEE 802.21 ha sido necesaria la implementación de un plugin que requiere que se tenga que volver a compilar la aplicación para su funcionamiento (Anexo 3).

En el Router y en el Home Agent se ha instalado el demonio radvd (router advertisement daemon) [23].

Además en el MN y en el CN se ha instalado la herramienta iperf [24], que permite el envío continuo de tramas UDP entre ambos nodos.

5.2 Configuración de los nodos de la red

5.2.1 Mobile Node

El ordenador portátil que actúa como Mobile Node posee dos tarjetas de red Wi-Fi (eth1 y wlan0).

La implementación de Mobile IPv6 se lleva a cabo mediante la instalación del parche USAGI Mobile IPv6 para Linux descrita en el Anexo 2. El contenido del archivo de configuración necesario para que este nodo actúe como Mobile Node es el siguiente (mip6d-mn.conf):

```
NodeConfig MN;

DebugLevel 10;

UseMnHaIPsec disabled;
DoRouteOptimizationMN disabled;
DoRouteOptimizationCN disabled;
```

```
Interface "eth0";
Interface "eth1";
Interface "wlan0";
MnHomeLink "eth0" {
        HomeAddress 2001:720:410:1000::3/64;
        HomeAgentAddress 2001:720:410:1000::2;
}
```

Para lanzar los procesos que hacen que este nodo tenga la funcionalidad de MN se ha desarrollado el siguiente script (mipv6-mn.sh):

```
#!/bin/sh
set -e

PATH=${PATH:+$PATH:}/usr/local/sbin:/usr/sbin:/sbin
mip6d_conf=/home/alberto/Escritorio/mip6d-mn.conf


case "$1" in
    start)

      # Ejecutar mip6d
      mip6d -c $mip6d_conf
        echo "Iniciando MIPv6 MN."
      ;;
    stop)
      killall mip6d

        echo "Deteniendo MIPv6 MN."
      ;;

    *)
      echo "Uso: ${0##*/} {start|stop}"
      exit 1
      ;;
esac

exit 0
```

Cuyo uso es el siguiente:

- Para arrancar el Mobile Node: ./mipv6-mn.sh start
- Para detener el Mobile Node: ./mipv6-mn.sh stop

Para el funcionamiento de IEEE 802.21 es necesario que en este nodo residan los ejecutables de los programas desarrollados en este proyecto y que se obtienen a partir de los siguientes archivos de código fuente:

- MIH_Protocol.h
- Mobile_Node.c

Además es necesario instalar el plugin de Wireshark desarrollado para la correcta decodificación de los mensajes del protocolo MIH. Para ello se necesita el archivo packet-mih.c y seguir los pasos descritos en el Anexo 3.

En este nodo es necesario instalar la aplicación iperf que permite el envío continuo de tramas UDP. Este nodo actúa como servidor respecto a esta aplicación, es

decir, es el encargado de procesar las tramas enviadas desde el Correspondent Node. Para lanzar la aplicación como servidor:

- iperf –s –u –V

El MN debe realizar handovers entre las dos redes. Para realizar estos handovers es necesario levantar el nuevo interfaz y conectarse al nuevo punto de acceso, y posteriormente apagar el interfaz que se ha dejado de usar. Con este propósito se han desarrollado cuatro scripts que llevan a cabo estas acciones. Dos de ellos se utilizan para el handover usando IEEE 802.21 y los otros dos para el handover sin IEEE802.21.

- Handover con IEEE802.21

Para el handover desde la network_a a la network_b se define el siguiente script (handover_802_21_atob):

```
#!/bin/sh

echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
echo 0 > /proc/sys/net/ipv6/conf/all/dad_transmits
echo 0 > /proc/sys/net/ipv6/conf/all/accept_dad
echo 0 > /proc/sys/net/ipv6/conf/all/router_solicitations
echo 0 > /proc/sys/net/ipv6/conf/wlan0/autoconf
echo 0 > /proc/sys/net/ipv6/conf/wlan0/dad_transmits
echo 0 > /proc/sys/net/ipv6/conf/wlan0/accept_dad
echo 0 > /proc/sys/net/ipv6/conf/wlan0/router_solicitations
ifconfig wlan0 up
echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
echo 0 > /proc/sys/net/ipv6/conf/all/dad_transmits
echo 0 > /proc/sys/net/ipv6/conf/all/accept_dad
echo 0 > /proc/sys/net/ipv6/conf/all/router_solicitations
echo 0 > /proc/sys/net/ipv6/conf/wlan0/autoconf
echo 0 > /proc/sys/net/ipv6/conf/wlan0/dad_transmits
echo 0 > /proc/sys/net/ipv6/conf/wlan0/accept_dad
echo 0 > /proc/sys/net/ipv6/conf/wlan0/router_solicitations
ifconfig wlan0 add 2001:720:410:1002:209:5bff:fee8:4648/64
ifconfig wlan0 add fe80::209:5bff:fee8:4648/64
route -A inet6 add default gw fe80::2e0:4cff:fea1:bc33 wlan0
iwconfig wlan0 essid network_b
sendip -p ipv6 -6l 64 -6n 0x3a -6h 255 -6s fe80::2e0:4cff:fea1:bc33 -
6d ff02::1 -
d0x8600b19c4000001e0000000000000000030440e000278d0000093a8000000000200
1072004101002000000000000000001010100e04ca1bc330701000000002710 -v
fe80::209:5bff:fee8:4648
ifconfig eth1 down

exit 0
```

Se configura el interfaz wlan0 con la dirección IPv6 perteneciente a la network_b y que representa la CoA del MN. También se configura la dirección del router de dicha red.

El mensaje IP enviado se corresponde con un router advertisement y se utiliza para que el parche de MIPv6 utilizado funcione correctamente cuando se trata de simular el comportamiento con IEEE 802.21, es decir, sin el envío por parte de la red de router advertisement de forma automática.

Para el handover desde la network_b a la network_a se define el siguiente script (handover_802_21_btoa):

```
#!/bin/sh

echo 0 > /proc/sys/net/ipv6/conf/all/router_solicitations
echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
echo 0 > /proc/sys/net/ipv6/conf/all/accept_dad
echo 0 > /proc/sys/net/ipv6/conf/all/dad_transmits
echo 0 > /proc/sys/net/ipv6/conf/eth1/autoconf
echo 0 > /proc/sys/net/ipv6/conf/eth1/dad_transmits
echo 0 > /proc/sys/net/ipv6/conf/eth1/accept_dad
echo 0 > /proc/sys/net/ipv6/conf/eth1/router_solicitations
ifconfig eth1 up
echo 0 > /proc/sys/net/ipv6/conf/all/router_solicitations
echo 0 > /proc/sys/net/ipv6/conf/all/autoconf
echo 0 > /proc/sys/net/ipv6/conf/all/dad_transmits
echo 0 > /proc/sys/net/ipv6/conf/all/accept_dad
echo 0 > /proc/sys/net/ipv6/conf/eth1/autoconf
echo 0 > /proc/sys/net/ipv6/conf/eth1/dad_transmits
echo 0 > /proc/sys/net/ipv6/conf/eth1/accept_dad
echo 0 > /proc/sys/net/ipv6/co#iwconfig eth1 essid network_a
ifconfig eth1 add 2001:720:410:1001:213:ceff:fe63:5cb9/64
ifconfig eth1 add fe80::213:ceff:fe63:5cb9/64
route -A inet6 add default gw fe80::2c0:26ff:fea3:7d8e eth1
iwconfig eth1 essid network_a
sendip -p ipv6 -6l 64 -6n 0x3a -6h 255 -6s fe80::2c0:26ff:fea3:7d8e -
6d ff02::1 -
d0x86007b244000001e0000000000000000030440e000278d0000093a8000000000200
1072004101001000000000000000001010100c026a37d8e0701000000002710 -v
fe80::213:ceff:fe63:5cb9
ifconfig wlan0 down

exit 0
```

Se configura el interfaz eth1 con la dirección IPv6 perteneciente a la networka_a y que representa la CoA del MN. También se configura la dirección del router de dicha red.

El mensaje IP enviado se corresponde con un router advertisement y se utiliza para que el parche de MIPv6 utilizado funcione correctamente cuando se trata de simular el comportamiento con IEEE 802.21, es decir, sin el envío por parte de la red de router advertisement de forma automática.

- Handover sin IEEE802.21

Para el handover desde la network_a a la network_b se define el siguiente script (handover_atob):

```
#!/bin/sh

ifconfig wlan0 up
iwconfig wlan0 essid network_b
ifconfig eth1 down

exit 0
```

Para el handover desde la network_a a la network_b se define el siguiente script (handover_atob):

```
#!/bin/sh

ifconfig eth1 up
iwconfig eth1 essid network_a
ifconfig wlan0 down

exit 0
```

5.2.2 Router

El PC de sobremesa que actúa como Router (PC 1) posee cuatro tarjetas de red (eth0, eth1, eth2 y eth3). Es necesario asignar una dirección IPv6 a cada uno de estos interfaces y añadir una ruta a la tabla de rutas para el correcto funcionamiento de Mobile IPv6. Para ello hay que modificar el fichero /etc/netowrk/interfaces de tal forma que en arranque se asigne a cada interfaz su dirección IPv6 correspondiente y se defina la ruta necesaria para Mobile IPv6. El contenido de este archivo es el siguiente:

```
auto lo
iface lo inet loopback
iface lo inet6 loopback


auto eth0
iface eth0 inet static
      address 10.0.0.1
      netmask 255.255.255.0


iface eth0 inet6 static
      address 2001:720:410:1000::1
      netmask 64
      route -A inet6 add 2001:720:410:1000::/64 gw
      2001:720:410:1000::2

auto eth1
iface eth1 inet static
      address 10.0.1.1
      netmask 255.255.255.0

iface eth1 inet6 static
      address 2001:720:410:1001::1
      netmask 64


auto eth2
iface eth2 inet static
      address 10.0.2.1
      netmask 255.255.255.0

iface eth2 inet6 static
      address 2001:720:410:1002::1
      netmask 64
```

```
auto eth3
iface eth3 inet static
      address 10.0.3.1
      netmask 255.255.255.0

iface eth3 inet6 static
      address 2001:720:410:1003::1
      netmask 64
```

Los nodos conectados a cada uno de estos interfaces son:

- eth0: Home Agent

- eth1: Punto de acceso 1 (network_a)

- eth2: Punto de acceso 2 (network_b)

- eth3: Correspondent Node

La ruta definida en este archivo es necesaria para el correcto enrutamiento de los paquetes cuando se está usando Mobile IPv6, ya que de esta forma, los paquetes dirigidos al MN son procesados por el Home Agent.

Cuando no se usa IEEE 802.21, es necesario usar el demonio radvd para que el MN pueda configurar una dirección IPv6 válida, a partir del prefijo anunciado, cuando se conecta a una red. Este demonio se encarga de enviar periódicamente los mensajes router advertisement por los interfaces (eth1 y eth2, en este caso) configurados para ello, y de responder a los mensajes router solicitations enviados por el MN.

El contenido del fichero de configuración para radvd (radvd.conf) en este caso es el siguiente:

```
interface eth1
{
        AdvSendAdvert on;
        AdvIntervalOpt on;

        MaxRtrAdvInterval 0.5;
        MinRtrAdvInterval 0.03;
        MinDelayBetweenRAs 0.03;
        AdvHomeAgentFlag off;

        prefix 2001:720:410:1001::1/64
        {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
        };
};

interface eth2
{
        AdvSendAdvert on;
        AdvIntervalOpt on;

        MaxRtrAdvInterval 0.5;
```

```
        MinRtrAdvInterval 0.03;
        MinDelayBetweenRAs 0.03;
        AdvHomeAgentFlag off;

        prefix 2001:720:410:1002::1/64
        {
        AdvOnLink on;
        AdvAutonomous on;
        AdvRouterAddr on;
        };
};
```

Donde se observa que por cada uno de los interfaces, y a través de los puntos de acceso, se anuncian dos prefijos de red distintos, que pertenecen a las dos redes que existen en el escenario de pruebas, y las que el MN se puede conectar y moverse entre ellas.

Para que el PC actúe como router es necesario habilitar la opción del reenvío de paquetes, bien mediante la variable /proc/sys/net/ipv6/conf/all/autoconf o modificando el archivo /etc/sysctl.conf. Además cuando no se usa IEEE 802.21 es necesario iniciar el demonio radvd, para lo que se ha desarrollado el siguiente script (mipv6-router.sh):

```
#!/bin/sh
set -e
radvd_conf=/home/alberto/Escritorio/radvd.conf

case "$1" in
    start)
      # Router. Habilitar el reenvio de paquetes
      echo 1 >/proc/sys/net/ipv6/conf/all/forwarding
      echo 0 >/proc/sys/net/ipv6/conf/all/autoconf
      echo 0 >/proc/sys/net/ipv6/conf/all/accept_ra
      echo 0 >/proc/sys/net/ipv6/conf/all/accept_redirects

      # RADVD
       radvd -C $radvd_conf

        echo "Iniciando MIPv6 RADVD"
      ;;
    stop)
      killall radvd

        echo "Deteniendo MIPv6 RADVD"
      ;;
    *)
      echo "Uso: ${0##*/} {start|stop}"
      exit 1
      ;;
esac

exit 0
```

Cuyo uso es el siguiente:

- Para arrancar el radvd: ./mipv6-router.sh start
- Para detener el radvd: ./mipv6-router.sh stop

Para el funcionamiento de IEEE 802.21 es necesario que en este nodo residan los ejecutables de los programas desarrollados en este proyecto y que se obtienen a partir de los siguientes archivos de código fuente:

- MIH_Protocol.h
- Serving_Network.c
- Candidate1.c
- Candidate2.c

Además es necesario instalar el plugin de Wireshark desarrollado para la correcta decodificación de los mensajes del protocolo MIH. Para ello se necesita el archivo packet-mih.c y seguir los pasos descritos en el Anexo 3.

5.2.3 Home Agent

El PC de sobremesa que actúa como Home Agent (PC 2) posee una tarjeta de red (eth0). Es necesario asignar una dirección IPv6 a este interfaz. Para ello hay que modificar el fichero /etc/netowrk/interfaces de tal forma que en arranque se asigne a ese interfaz su dirección IPv6 correspondiente. El contenido de este archivo es el siguiente:

```
auto lo
iface lo inet loopback
iface lo inet6 loopback


auto eth0
iface eth0 inet static
      address 10.0.0.2
      netmask 255.255.255.0


iface eth0 inet6 static
      address 2001:720:410:1000::2
      netmask 64
      gateway 2001:720:410:1000::1
```

En este nodo es necesario instalar el demonio radvd para permitir un correcto funcionamiento de Mobile IPv6. El archivo de configuración de este demonio es el siguiente (radvd.conf):

```
interface eth0
{
      AdvSendAdvert on;

      MinRtrAdvInterval 3;
      MaxRtrAdvInterval 10;

#
# Soporte Mobile IPv6
#
      AdvHomeAgentFlag on;

#
# Prefijo
#
      prefix 2001:720:410:1000::2/64
```

```
      {
            AdvOnLink on;
            AdvAutonomous on;
            AdvRouterAddr on;
      };
};
```

La implementación de Mobile IPv6 se lleva a cabo mediante la instalación del parche USAGI Mobile IPv6 para Linux descrita en el Anexo 2. El contenido del archivo de configuración necesario para que este nodo actúe como Home Agent es el siguiente (mip6d-ha.conf):

```
NodeConfig HA;
DebugLevel 0;
Interface "eth0";
UseMnHaIPsec disabled;
DoRouteOptimizationMN  disabled;
DoRouteOptimizationCN disabled;
BindingAclPolicy 2001:720:410:1000::3 allow;
```

Para lanzar los procesos que hacen que este nodo tenga la funcionalidad de Home Agent se ha desarrollado el siguiente script (mipv6-ha.sh):

```
#!/bin/sh
set -e

PATH=/usr/local/sbin:/usr/sbin:/sbin:${PATH}
mip6d_conf=/home2/aascension/Escritorio/mip6d-ha.conf
radvd_conf=/home2/aascension/Escritorio/radvd.conf


case "$1" in
    start)
      # Router
      echo 1 >/proc/sys/net/ipv6/conf/all/forwarding
      echo 0 >/proc/sys/net/ipv6/conf/all/autoconf
      echo 0 >/proc/sys/net/ipv6/conf/all/accept_ra
      echo 0 >/proc/sys/net/ipv6/conf/all/accept_redirects


      # ejecutar mip6d
      mip6d -c $mip6d_conf
      # RADVD
      radvd -C $radvd_conf

        echo "Iniciando MIPv6 HA."
      ;;
    stop)
      killall radvd
      killall mip6d

        echo "Deteniendo MIPv6 HA."
      ;;
    *)
      echo "Uso: ${0##*/} {start|stop}"
      exit 1
      ;;
esac
exit 0
```

Cuyo uso es el siguiente:

- Para arrancar el Home Agent: ./mipv6-ha.sh start
- Para detener el Home Agent: ./mipv6-ha.sh stop

Además es necesario instalar el plugin de Wireshark desarrollado para la correcta decodificación de los mensajes del protocolo MIH. Para ello se necesita el archivo packet-mih.c y seguir los pasos descritos en el Anexo 3.

5.2.4 Correspondent Node

El PC de sobremesa que actúa como Correspondent Node (PC 3) posee una tarjeta de red (eth0). Es necesario asignar una dirección IPv6 a este interfaz. Para ello hay que modificar el fichero /etc/netowrk/interfaces de tal forma que en arranque se asigne a ese interfaz su dirección IPv6 correspondiente. El contenido de este archivo es el siguiente:

```
auto lo
iface lo inet loopback
iface lo inet6 loopback

auto eth0
iface eth0 inet static
      address 10.0.3.2
      netmask     255.255.255.0
      gateway     10.0.3.1

iface eth0 inet6 static
      address 2001:720:410:1003::2
      netmask     64
      gateway     2001:720:410:1003::1
```

La implementación de Mobile IPv6 se lleva a cabo mediante la instalación del parche USAGI Mobile IPv6 para Linux descrita en el Anexo 2. El contenido del archivo de configuración necesario para que este nodo actúe como Correspondent Node es el siguiente (mip6d-mn.conf):

```
NodeConfig CN;

DebugLevel 10;

DoRouteOptimizationCN disabled;
DoRouteOptimizationMN disabled;
```

Para lanzar el proceso que hace que este nodo tenga la funcionalidad de Correspondent Node se ha desarrollado el siguiente script (mipv6-cn.sh):

```
#!/bin/sh
set -e

PATH=/usr/local/sbin:/usr/sbin:/sbin:${PATH}
mip6d_conf=/home2/aascension/Escritorio/mip6d-cn.conf
```

```
case "$1" in
    start)
      # Ejecutar mip6d
      mip6d -c $mip6d_conf

        echo "Iniciando MIPv6 CN."
      ;;
    stop)
      killall mip6d

        echo "Deteniendo MIPv6 CN."
      ;;
    *)
      echo "Uso: ${0##*/} {start|stop}"
      exit 1
      ;;
esac

exit 0
```

Cuyo uso es el siguiente:

- Para arrancar el Correspondent Node: ./mipv6-cn.sh start
- Para detener el Correspondent Node: ./mipv6-cn.sh stop

En este nodo es necesario instalar la aplicación iperf que permite el envío continuo de tramas UDP. Este nodo actúa como cliente respecto a esta aplicación, es decir, es el encargado de enviar tramas continuamente al MN. Para lanzar la aplicación como cliente:

- iperf –c –u –V `2001:720:410:1000::3 –t 60`

Además es necesario instalar el plugin de Wireshark desarrollado para la correcta decodificación de los mensajes del protocolo MIH. Para ello se necesita el archivo packet-mih.c y seguir los pasos descritos en el Anexo 3.

5.2.5 Puntos de acceso (AP) Wi-Fi

Se dispone de dos routers Wi-Fi Linksys WRT54GL que hay que configurar como APs. Para ello hay que acceder a la página de configuración del router utilizando las instrucciones del fabricante. En la página de configuración hay que modificar los siguientes parámetros:

1.- En la pantalla Basic Setup → Network Setup modificar el campo Local IP Address Network, introduciendo la dirección IP asignada a este interfaz en relación a la red a la que pertenece, en este caso, 10.0.1.2 (AP1) y 10.0.2.2 (AP2) respectivamente para cada uno de los puntos de acceso.

2.- En la pantalla Basic Setup → Network Setup deshabilitar el DHCP.

3.- En la pantalla Wireless → Basic Wireless Settings modificar los siguientes parámetros:

● Wireless Network Name (SSID): Nombre de la red. En este caso se introducirá network_a (AP1) y network_b (AP2) respectivamente para cada uno de los puntos de acceso.

● Wireless Channel: Elegir dos canales separados para cada uno de los puntos de acceso. En este caso el canal 11 para el AP1 y el canal 8 para el AP2.

Cada uno de los APs tiene que estar conectado a un interfaz del router mediante un cable de red que va desde el interfaz correspondiente en el router a uno de los puertos Ethernet que posee el AP.

6. Resultados experimentales

En este apartado se muestran los resultados obtenidos en las pruebas realizadas sobre el escenario montado en el laboratorio (Figura 5.1).

Las pruebas consisten en el intercambio de paquetes UDP entre el Correspondent Node y el Mobile Node, para lo que se ha utilizado la herramienta iperf (apartado 5.5) actuando el Correspondent Node como cliente (fuente) y el Mobile Node como servidor (destino), durante el cual se lleva a cabo un handover entre los dos puntos de acceso (que representan dos redes diferentes).

El objetivo es medir el tiempo empleado en realizar el handover. Para ello se capturan las tramas recibidas en los interfaces del Mobile Node mediante la herramienta Wireshark (Anexo 3). A partir de ellas se calcula el tiempo empleado en el handover observando la diferencia de tiempos entre la última trama recibida por el interfaz fuente del handover (punto de acceso 2, network_b, interfaz wlan0) y la primera trama recibida por el interfaz destino del handover (punto de acceso 1, network_a, interfaz eth1).

Para comprobar la calidad del estimador, obtenido a partir de las capturas realizadas, de la media del tiempo empleado en el handover se ha recurrido a calcular el número de muestras necesarias para obtener un intervalo de confianza del 95% para la media (suponiendo que ésta se distribuye según una $N(\mu,\sigma)$) [25].

Se calcula el número de muestras, N, de tal forma que la diferencia entre la media estimada del tiempo empleado en el handover, X, y la media real μ, sea menor que un valor dado ε, es decir, $| X - \mu | < \varepsilon$ con una probabilidad $1- \alpha$ ($\alpha = 0.05$).

6.1 Handover iniciado por el Mobile Node

6.1.1 Rendimiento usando Router Advertisement

En este caso el handover se lleva a cabo sin la utilización de IEEE 802.21, de tal forma que no se tiene información previa acerca de la red donde se va a realizar el handover, es decir, es necesario obtener el prefijo de red para formar una CoA así como la dirección del nuevo router de acceso puesto que no se conoce de antemano. Esta información se obtiene de los mensajes router advertisement enviados por la nueva red.

Se han realizado 30 capturas del proceso de handover obteniéndose los resultados mostrados en la siguiente tabla (Tabla 6.1):

CAPTURA	TIEMPO HANDOVER (segundos)
1	1,734
2	2,168
3	2,334
4	1,793
5	2,127
6	2,022
7	2,022
8	2,041
9	2,012

10	1,445
11	2,374
12	1,971
13	2,23
14	1,815
15	1,762
16	1,807
17	2,258
18	2,362
19	2,258
20	2,248
21	1,834
22	1,771
23	1,938
24	2,184
25	1,853
26	1,817
27	1,964
28	2,112
29	2,086
30	1,87

Tabla 6.1: Tiempos empleados para el handover sin IEEE 802.1

La media del tiempo empleado para realizar el handover se obtiene a partir de los resultados mostrados en la Tabla 6.1:

Tiempo medio del handover sin IEEE 802.21= 60,712/ 30 = **2.0237 segundos**

Se calcula el número de muestras para que el tiempo medio empleado en realizar el handover esté dentro del intervalo de confianza de confianza deseado, en este caso, del 95%. De esta forma el número de muestras necesario es 15, por lo que el número de muestras tomado es suficiente para el nivel de confianza buscado.

En la siguiente figura (Figura 6.1) se muestran las tramas UDP recibidas en cada uno de los interfaces durante una transmisión entre fuente (CN) y destino (MN) durante la cual se ha realizado un handover entre las dos redes existentes.

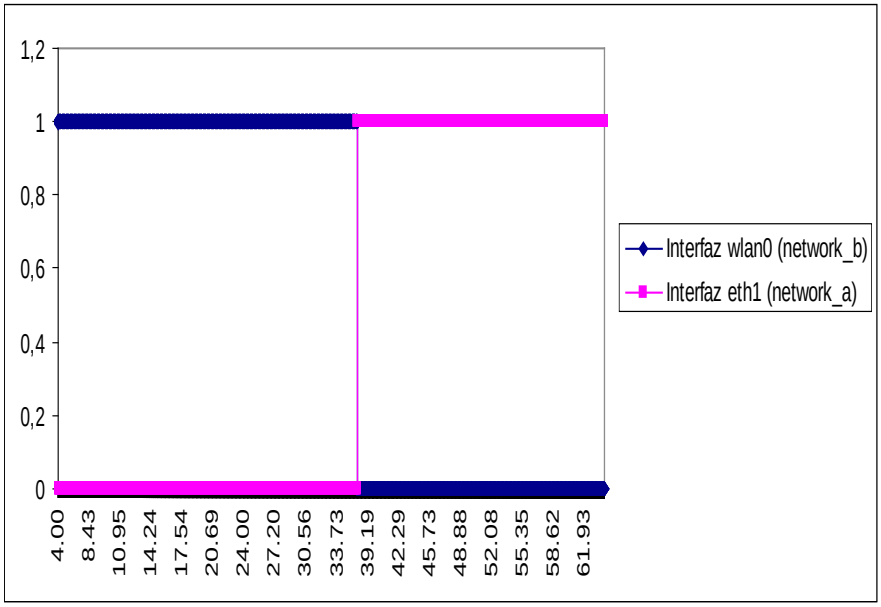

Figura 6.1: Recepción de tramas UDP durante el handover sin IEEE802.21

En la Figura 6.1, el eje de ordenadas representa el tiempo durante el cual se ha realizado el handover. El eje de abscisas representa la recepción o no recepción de una trama por parte de un interfaz, de tal forma que si un interfaz ha recibido una trama UDP su valor es 1 mientras que si no ha recibido trama su valor es 0.

En la siguiente figura (Figura 6.2) se muestran los instantes de tiempo, de forma más detallada, durante el cual se ha realizado el handover.

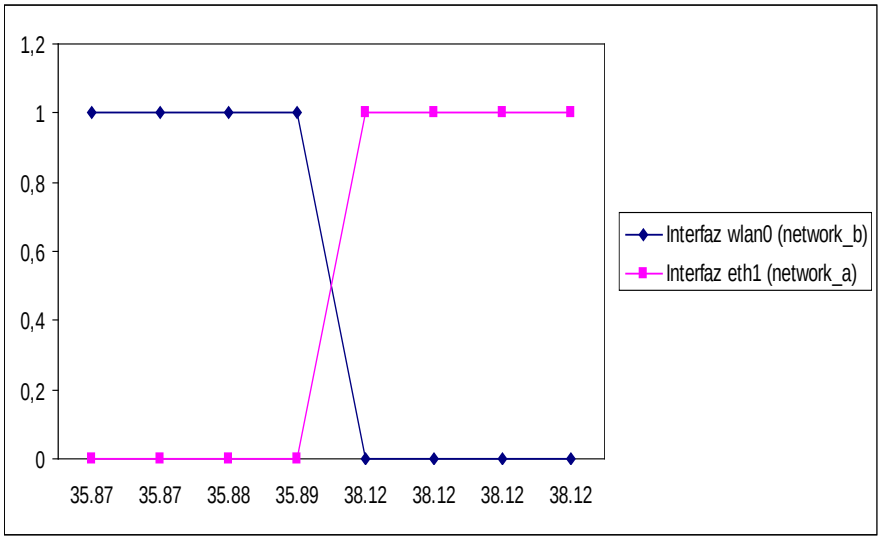

Figura 6.2: Detalle del instante del handover sin IEEE802.21

En la Figura 6.2 se observa el intervalo durante el cual se realiza el handover. Durante este proceso el MN deja de recibir tramas por el interfaz origen del handover (wlan0) ya que éste ha sido apagado y no es capaz de recibir tramas por el interfaz

destino (eth1) ya que es necesario construir una CoA válida, para lo cual es necesario procesar los mensajes de router advertisement que son enviados por parte de la nueva red. Una vez que el MN ha configurado su CoA manda el mensaje de Binding Update a su Home Agent para registrarla y poder volver a recibir tramas dirigidas a él. Estas tramas serán recibidas a través del nuevo interfaz (eth1) y el proceso de handover habrá finalizado.

En la siguiente figura (Figura 6.3) se muestran las tramas recibidas en el MN durante el proceso de handover:

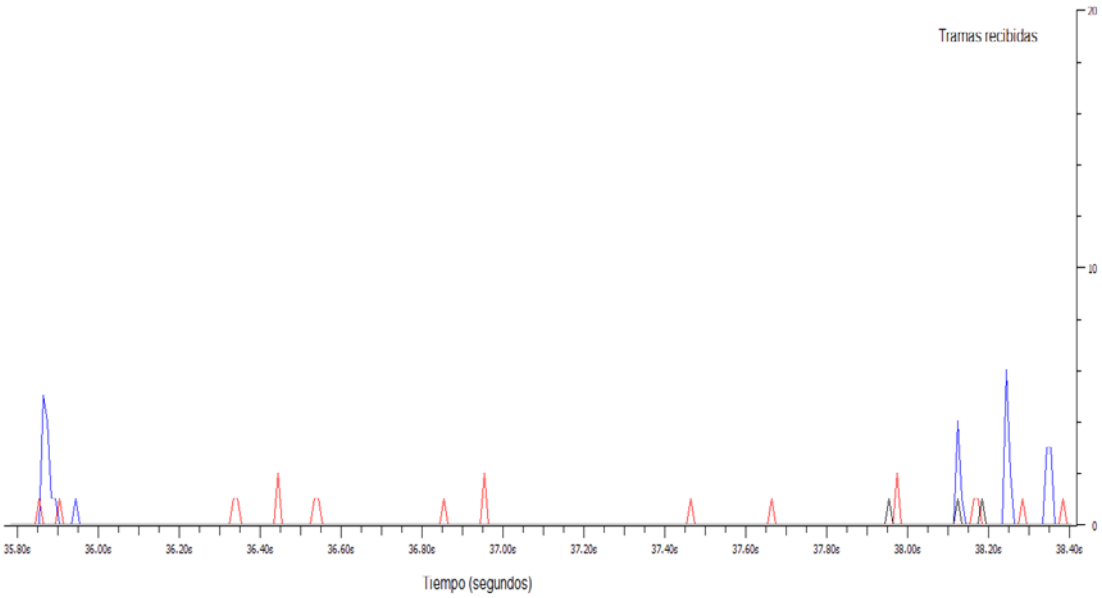

Figura 6.3: Mensajes intercambiados durante el handover sin IEEE802.21

En color azul se representan las tramas UDP recibidas en el MN. Se observa que durante el proceso de handover se dejan de recibir tramas procedentes del CN debido a que el enlace fuente del handover ha sido desconectado y el enlace destino no está configurado durante este tiempo para recibir tramas de la transacción en curso. Una vez que dicho enlace esta configurado se vuelven a recibir tramas UDP en el MN (por el nuevo interfaz).

En color rojo se representan los mensajes intercambiados entre el MN y la nueva red de acceso correspondientes a la configuración del nuevo interfaz. Estos mensajes son los router advertisement, neighbor solicitation y neighbor advertisement. Mediante estos mensajes es posible la configuración de una dirección IPv6 válida (CoA) en la nueva red.

En color negro se representan los mensajes correspondientes al protocolo Mobile IPv6. Estos mensajes son el binding update y el binding acknowlegement. Una vez que el MN registra la nueva CoA con su Home Agent mediante el envío del binding update, es capaz de volver a recibir las tramas enviadas por el CN.

6.1.2 Rendimiento usando IEEE802.21

En este caso el handover se lleva a cabo con la utilización de IEEE 802.21, de tal forma que se tiene información previa acerca de la red donde se va a realizar el handover, es decir, no es necesario obtener el prefijo de red para formar una CoA, ni la dirección del nuevo router de acceso puesto que se conoce de antemano. Esta información se obtiene de los mensajes de señalización pertenecientes al protocolo MIH intercambiados entre el MN y la Serving Network (y entre ésta y las redes candidatas) previas a la realización del handover y descritos en los apartados 3.5 y 3.6.

Se han realizado 30 capturas del proceso de handover obteniéndose los resultados mostrados en la siguiente tabla (Tabla 6.2):

CAPTURA	TIEMPO HANDOVER (segundos)
1	0,532
2	0,639
3	0,546
4	0,523
5	0,617
6	0,388
7	0,487
8	0,491
9	0,561
10	0,567
11	0,545
12	0,597
13	0,603
14	0,459
15	0,561
16	0,462
17	0,465
18	0,526
19	0,482
20	0,541
21	0,513
22	0,636
23	0,546
24	0,578
25	0,399
26	0,468
27	0,438
28	0,562
29	0,495
30	0,455

Tabla 6.2: Tiempos empleados para el handover con IEEE 802.1 (MN Initiated)

La media del tiempo empleado para realizar el handover se obtiene a partir de los resultados mostrados en la Tabla 6.2:

Tiempo medio del handover sin IEEE 802.21= 15,682/ 30 = **0.5227 segundos**

Se calcula el número de muestras para que el tiempo medio empleado en realizar el handover esté dentro del intervalo de confianza de confianza deseado, en este caso, del 95%. De esta forma el número de muestras necesario es 7, por lo que el número de muestras tomado es suficiente para el nivel de confianza buscado.

En la siguiente figura (Figura 6.4) se muestran las tramas UDP recibidas en cada uno de los interfaces durante una transmisión entre fuente (CN) y destino (MN) durante la cual se ha realizado un handover (utilizando IEEE 802.21) entre las dos redes existentes.

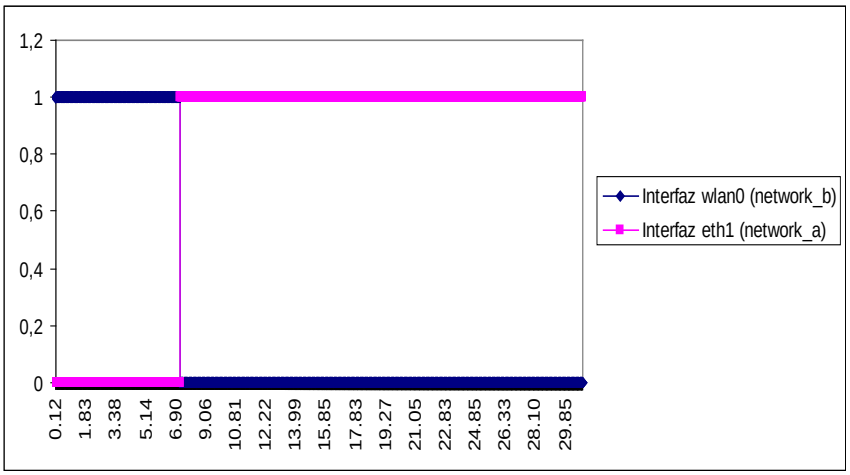

Figura 6.4: Recepción de tramas UDP durante el handover con IEEE802.21 (MN Initiated)

En la Figura 6.4, el eje de ordenadas representa el tiempo durante el cual se ha realizado el handover. El eje de abscisas representa la recepción o no recepción de una trama por parte de un interfaz, de tal forma que si un interfaz ha recibido una trama UDP su valor es 1 mientras que si no ha recibido trama su valor es 0.

En la siguiente figura (Figura 6.5) se muestran los instantes de tiempo, de forma más detallada, durante el cual se ha realizado el handover.

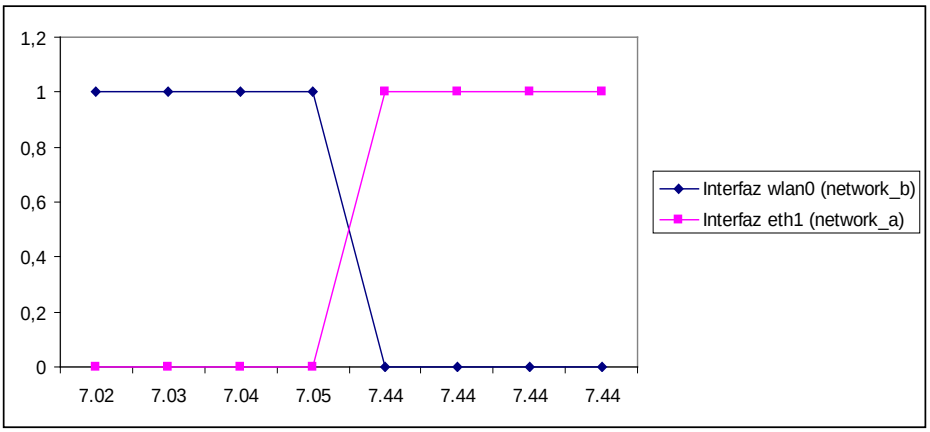

Figura 6.5: Detalle del instante del handover con IEEE802.21 (MN Initiated)

En la Figura 6.5 se observa el intervalo durante el cual se realiza el handover. Durante este proceso el MN deja de recibir tramas por el interfaz origen del handover (wlan0) ya que éste ha sido apagado y no es capaz de recibir tramas por el interfaz destino (eth1). En este caso el MN ya tiene configurada su CoA ya que antes de realizar el handover obtiene información para formarla a partir de los mensajes de señalización pertenecientes al protocolo MIH. El MN envía el binding update a su Home Agent, para registrar dicha CoA y poder volver a recibir tramas dirigidas a él. Estas tramas serán recibidas a través del nuevo interfaz (eth1) y el proceso de handover habrá finalizado.

En la siguiente figura (Figura 6.6) se muestran las tramas recibidas en el MN durante el proceso de handover:

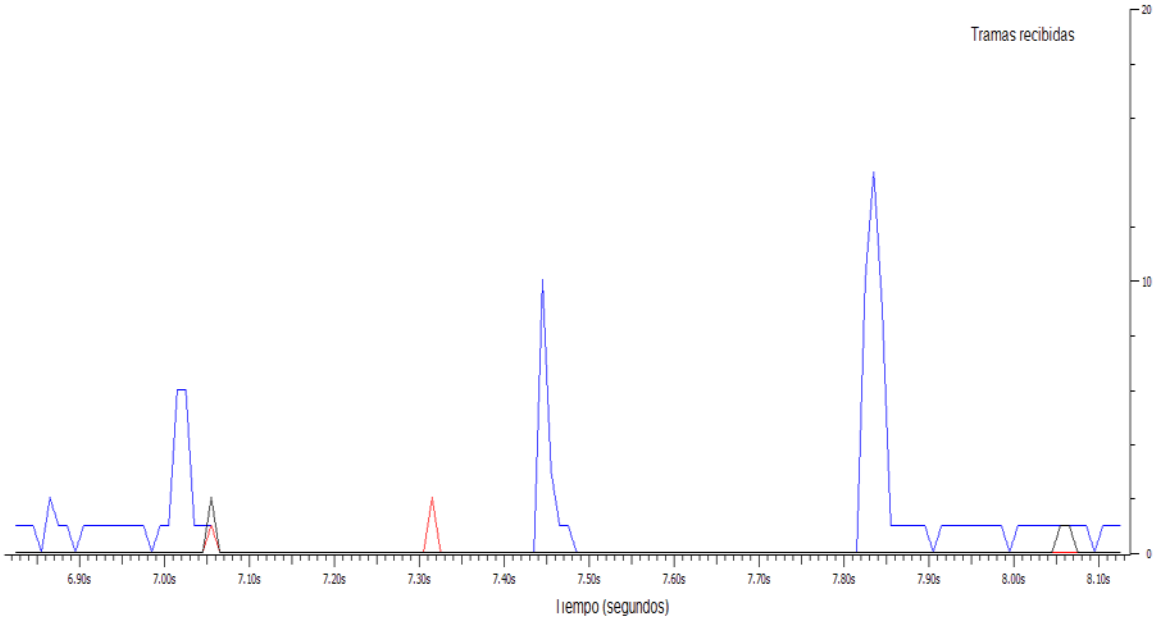

Figura 6.6: Mensajes intercambiados durante el handover con IEEE802.21 (MN Initiated)

En color azul se representan las tramas UDP recibidas en el MN. Se observa que durante el proceso de handover se dejan de recibir tramas procedentes del CN debido a que el enlace fuente del handover ha sido desconectado y el enlace destino no está configurado durante este tiempo para recibir tramas de la transacción en curso. Una vez que dicho enlace esta configurado se vuelven a recibir tramas UDP en el MN (por el nuevo interfaz).

En color rojo se representan los mensajes intercambiados entre el MN y la nueva red de acceso correspondientes a la configuración del nuevo interfaz. Estos mensajes son los neighbor solicitation y neighbor advertisement. El router advertisement observado se utiliza para que el parche de MIPv6 utilizado funcione correctamente cuando se trata de simular el comportamiento con IEEE 802.21, es decir, sin el envío por parte de la red de router advertisement de forma automática.

En color negro se representan los mensajes correspondientes al protocolo Mobile IPv6. Estos mensajes son el binding update y el binding acknowlegement. Una vez que

el MN registra la nueva CoA con su Home Agent mediante el envío del binding update, es capaz de volver a recibir las trams enviadas por el CN.

A continuación se muestra un ejemplo de un mensaje de señalización del protocolo MIH (MIH_MN_HO_Candidate Request) utilizado para la realización del handover iniciado por el Mobile Node y capturado con el plugin creado para la herramienta Wireshark que decodifica los distintos campos el mensaje:

```
No.     Time        Source                Destination
Protocol Info
      1 0.000000    2001:720:410:1000::3  2001:720:410:1000::1  MIH
MIH_MN_HO_Candidate_Query Request

Frame 1 (195 bytes on wire, 195 bytes captured)
    Arrival Time: Jan 20, 2010 12:02:09.801834000
    [Time delta from previous captured frame: 0.000000000 seconds]
    [Time delta from previous displayed frame: 0.000000000 seconds]
    [Time since reference or first frame: 0.000000000 seconds]
    Frame Number: 1
    Frame Length: 195 bytes
    Capture Length: 195 bytes
    [Frame is marked: False]
    [Protocols in frame: sll:ipv6:udp:mih]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
Linux cooked capture
    Packet type: Sent by us (4)
    Link-layer address type: 769
    Link-layer address length: 0
    Source: <MISSING>
    Protocol: IPv6 (0x86dd)
Internet Protocol Version 6
    0110 .... = Version: 6
        [0110 .... = This field makes the filter "ip.version == 6"
possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class:
0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 139
    Next header: UDP (0x11)
    Hop limit: 64
    Source: 2001:720:410:1000::3 (2001:720:410:1000::3)
    Destination: 2001:720:410:1000::1 (2001:720:410:1000::1)
User Datagram Protocol, Src Port: 21100 (21100), Dst Port: 20020
(20020)
    Source port: 21100 (21100)
    Destination port: 20020 (20020)
    Length: 139
    Checksum: 0x90fd [validation disabled]
        [Good Checksum: False]
        [Bad Checksum: False]
MIH Protocol
    MIH Header
        .... 1... = Bit ACK Req: 1
        .... .0.. = Bit ACK Rsp: 0
        Message ID: MIH_MN_HO_Candidate_Query Request (0x3405)
            0011 .... = SID: Command Service  (3)
```

```
            .... 01.. = Opcode: Request (1)
            .... ..00 0000 0101 = AID: MIH_MN_HO_Candidate_Query (5)
        Payload Length: 123
    MIH Payload
        Source Identifier: 10.0.0.1 (10.0.0.1)
        Destination Identifier: 10.0.0.2 (10.0.0.2)
        Source Link Identifier
            Link Type: IEEE 802.11 (19)
            MN Link Addr: Netgear_e8:46:48 (00:09:5b:e8:46:48)
            PoA Link Addr: Cisco-Li_a4:37:f0 (00:1c:10:a4:37:f0)
        Candidate Link List
            Link Type: IEEE 802.11 (19)
            MN Link Addr: IntelCor_63:5c:b9 (00:13:ce:63:5c:b9)
            PoA Link Addr: Cisco-Li_a4:37:e4 (00:1c:10:a4:37:e4)
            PoA Link Addr: 3a:3b:3c:3d:3e:3f (3a:3b:3c:3d:3e:3f)
        QoS Resource Requirements
            Num Cos Types: 10
            Min Pk Tx Delay
                Cos ID: 1
                Packet delay/loss rate: 4113
            Avg Pk Tx Delay
                Cos ID: 2
                Packet delay/loss rate: 8225
            Max Pk Tx Delay
                Cos ID: 3
                Packet delay/loss rate: 12337
            Pk Delay Jitter
                Cos ID: 4
                Packet delay/loss rate: 16449
            Pk Loss Rate
                Cos ID: 5
                Packet delay/loss rate: 20561
        IP Configuration Methods
            .... .1.. = IPv6 Stateless address configuration: 0x01
        Access      Router      Address:      fe80::2e0:4cff:fea1:bc33
(fe80::2e0:4cff:fea1:bc33)
```

6.1.3 Comparativa Router Advertisement- IEEE 802.21

En la siguiente figura (Figura 6.7) se representa la comparativa de los tiempos de handover utilizando IEEE 8021.21 y utilizando router advertisement para las pruebas realizadas:

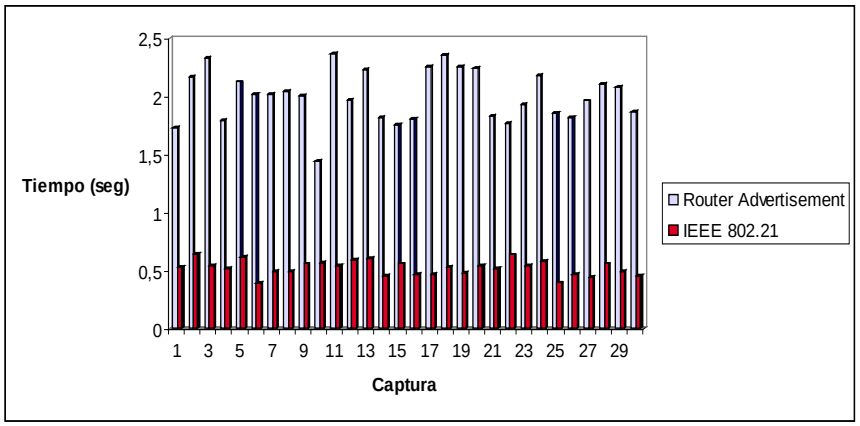

Figura 6.7: Comparativa IEEE 802.21 – Router Advertisement (MN Initiated)

Como se observa en la gráfica el tiempo empleado para el handover usando IEEE 802.21 es bastante inferior al empleado usando router Advertisement ya que con el protocolo MIH se puede construir la CoA y obtener la dirección del router de acceso de la nueva red antes de realizar el handover mediante la información intercambiada con los mensajes de señalización IEEE 802.21. Por el contrario sin IEEE 802.21 es necesario recibir los router advertisement y a partir del prefijo indicado construir la CoA correspondiente.

A partir de la media de los tiempos empleados se observa que con la utilización del protocolo MIH se reduce aproximadamente un 75% el tiempo empleado en el handover, lo que supone una importante mejora.

6.2 Handover iniciado por la Red

6.2.1 Rendimiento usando Router Advertisement

En este caso el handover es llevado a cabo exclusivamente por la entidad de nivel superior sin la intervención del protocolo MIH por lo que no es posible realizar la simulación ya que corresponde a una modificación del parche usado y no es el objetivo de este proyecto. Además el proceso de handover lo controla el MN ya que es el encargado de encender el nuevo enlace y apagar el antiguo por lo que las pruebas descritas en el apartado 6.1.1 pueden ser tomadas como referencia en este caso y usadas en el caso del handover iniciado por la Red usando router advertisement.

De esta forma se pude tomar el siguiente resultado:

Tiempo medio del handover sin IEEE 802.21= 2.0237 segundos

6.2.2 Rendimiento usando IEEE802.21

En este caso el handover se lleva a cabo con la utilización de IEEE 802.21, de tal forma que se tiene información previa acerca de la red donde se va a realizar el handover, es decir, no es necesario obtener el prefijo de red para formar una CoA, ni la dirección del nuevo router de acceso puesto que se conoce de antemano. Esta información se obtiene de los mensajes de señalización pertenecientes al protocolo MIH intercambiados entre el MN y la Serving Network (y entre ésta y las redes candidatas) previas a la realización del handover y descritos en los apartados 3.5 y 3.6.

Aquí el handover es iniciado por la Serving Network, a diferencia que el caso descrito en el apartado 6.1.2, donde el handover es iniciado por el MN. La Serving Network es la que solicita la información al MISS e inicia el proceso del handover.

Se han realizado 30 capturas del proceso de handover obteniéndose los resultados mostrados en la siguiente tabla (Tabla 6.3):

CAPTURA	TIEMPO HANDOVER (segundos)
1	0,498
2	0,504
3	0,587
4	0,515
5	0,612
6	0,401
7	0,445
8	0,496
9	0,498
10	0,534
11	0,589
12	0,498
13	0,534
14	0,521
15	0,601
16	0,471
17	0,459
18	0,534
19	0,497
20	0,467
21	0,589
22	0,506
23	0,478
24	0,602
25	0,401
26	0,568
27	0,489
28	0,465
29	0,595
30	0,503

Tabla 6.3: Tiempos empleados para el handover con IEEE 802.1 (Network initiated)

La media del tiempo empleado para realizar el handover se obtiene a partir de los resultados mostrados en la Tabla 6.3:

Tiempo medio del handover sin IEEE 802.21= 15,457/ 30 = **0,5152 segundos**

Se calcula el número de muestras para que el tiempo medio empleado en realizar el handover esté dentro del intervalo de confianza de confianza deseado, en este caso, del 95%. De esta forma el número de muestras necesario es 6, por lo que el número de muestras tomado es suficiente para el nivel de confianza buscado.

En la siguiente figura (Figura 6.8) se muestran las tramas UDP recibidas en cada uno de los interfaces durante una transmisión entre fuente (CN) y destino (MN) durante la cual se ha realizado un handover, iniciado por la Serving Network, y utilizando IEEE 802.21, entre las dos redes existentes.

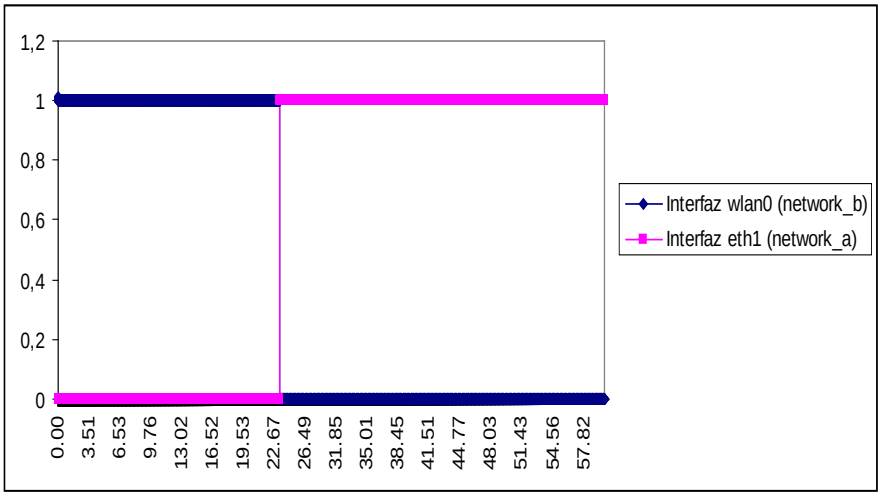

Figura 6.8: Recepción de tramas UDP durante el handover con IEEE802.21 (Network Initiated)

En la Figura 6.8, el eje de ordenadas representa el tiempo durante el cual se ha realizado el handover. El eje de abscisas representa la recepción o no recepción de una trama por parte de un interfaz, de tal forma que si un interfaz ha recibido una trama UDP su valor es 1 mientras que si no ha recibido trama su valor es 0.

En la siguiente figura (Figura 6.9) se muestran los instantes de tiempo, de forma más detallada, durante el cual se ha realizado el handover.

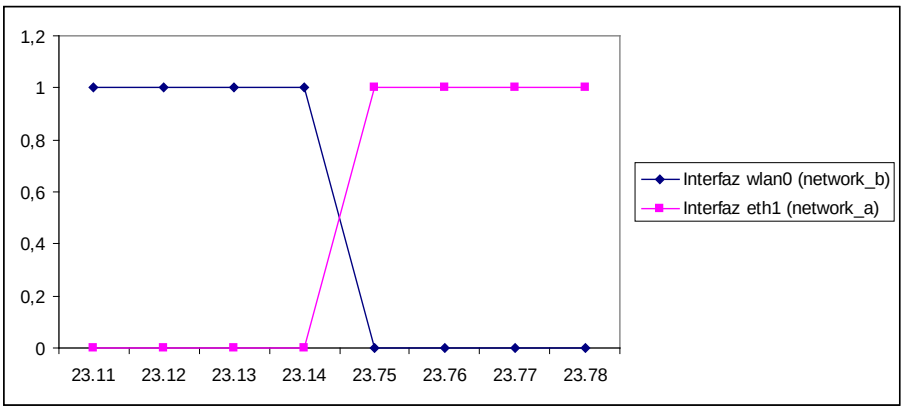

Figura 6.9: Detalle del instante del handover con IEEE802.21 (Network Initiated)

En la Figura 6.9 se observa el intervalo durante el cual se realiza el handover. Durante este proceso el MN deja de recibir tramas por el interfaz origen del handover (wlan0) ya que éste ha sido apagado y no es capaz de recibir tramas por el interfaz destino (eth1). En este caso el MN ya tiene configurada su CoA ya que antes de realizar el handover obtiene información para formarla a partir de los mensajes de señalización pertenecientes al protocolo MIH. El MN envía el binding update a su Home Agent, para registrar dicha CoA y poder volver a recibir tramas dirigidas a él. Estas tramas serán recibidas a través del nuevo interfaz (eth1) y el proceso de handover habrá finalizado.

En la siguiente figura (Figura 6.10) se muestran las tramas recibidas en el MN durante el proceso de handover:

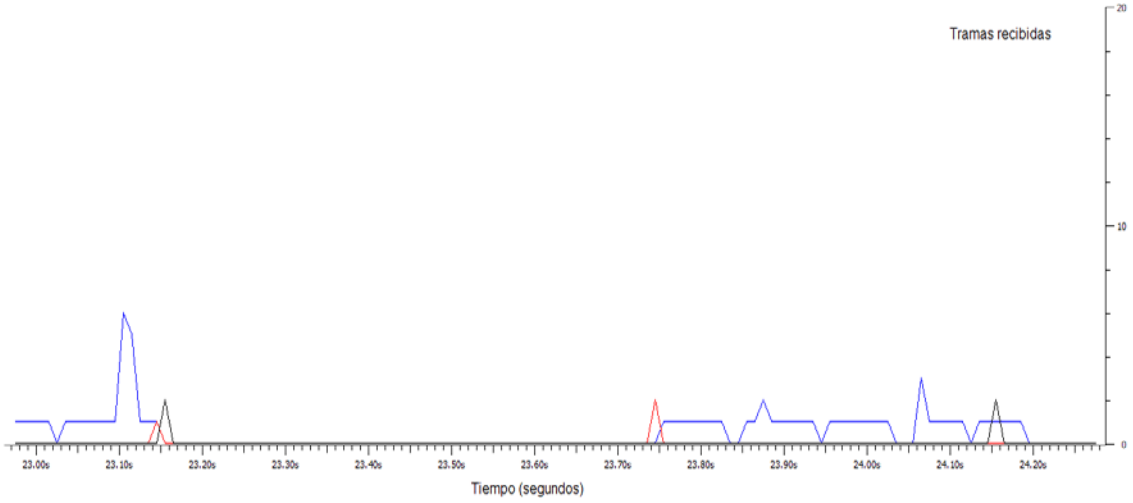

Figura 6.10: Mensajes intercambiados durante el handover con IEEE802.21 (Network Initiated)

En color azul se representan las tramas UDP recibidas en el MN. Se observa que durante el proceso de handover se dejan de recibir tramas procedentes del CN debido a que el enlace fuente del handover ha sido desconectado y el enlace destino no está configurado durante este tiempo para recibir tramas de la transacción en curso. Una vez que dicho enlace esta configurado se vuelven a recibir tramas UDP en el MN (por el nuevo interfaz).

En color rojo se representan los mensajes intercambiados entre el MN y la nueva red de acceso correspondientes a la configuración del nuevo interfaz. Estos mensajes son los neighbor solicitation y neighbor advertisement. El router advertisement observado se utiliza para que el parche de MIPv6 utilizado funcione correctamente cuando se trata de simular el comportamiento con IEEE 802.21, es decir, sin el envío por parte de la red de router advertisement de forma automática.

En color negro se representan los mensajes correspondientes al protocolo Mobile IPv6. Estos mensajes son el binding update y el binding acknowlegement. Una vez que el MN registra la nueva CoA con su Home Agent mediante el envío del binding update, es capaz de volver a recibir las trams enviadas por el CN.

A continuación se muestra un ejemplo de un mensaje de señalización del protocolo MIH (MIH_Net_HO_Commit Request) utilizado para la realización del handover iniciado por la Red y capturado con el plugin creado para la herramienta Wireshark que decodifica los distintos campos el mensaje:

```
Frame 11 (206 bytes on wire, 206 bytes captured)
    Arrival Time: Jan 20, 2010 12:20:33.152503000
    [Time delta from previous captured frame: 5.052255000 seconds]
    [Time delta from previous displayed frame: 5.052255000 seconds]
    [Time since reference or first frame: 5.063025000 seconds]
```

```
    Frame Number: 11
    Frame Length: 206 bytes
    Capture Length: 206 bytes
    [Frame is marked: False]
    [Protocols in frame: sll:ipv6:ipv6:udp:mih]
    [Coloring Rule Name: UDP]
    [Coloring Rule String: udp]
Linux cooked capture
    Packet type: Unicast to us (0)
    Link-layer address type: 1
    Link-layer address length: 6
    Source: RealtekS_a1:bc:33 (00:e0:4c:a1:bc:33)
    Protocol: IPv6 (0x86dd)
Internet Protocol Version 6
    0110 .... = Version: 6
        [0110 .... = This field makes the filter "ip.version == 6"
possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class:
0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 150
    Next header: IPv6 (0x29)
    Hop limit: 63
    Source: 2001:720:410:1000::2 (2001:720:410:1000::2)
    Destination: 2001:720:410:1002:209:5bff:fee8:4648
(2001:720:410:1002:209:5bff:fee8:4648)
Internet Protocol Version 6
    0110 .... = Version: 6
        [0110 .... = This field makes the filter "ip.version == 6"
possible: 6]
    .... 0000 0000 .... .... .... .... .... = Traffic class:
0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 110
    Next header: UDP (0x11)
    Hop limit: 63
    Source: 2001:720:410:1000::1 (2001:720:410:1000::1)
    Destination: 2001:720:410:1000::3 (2001:720:410:1000::3)
User Datagram Protocol, Src Port: 20020 (20020), Dst Port: 21100
(21100)
    Source port: 20020 (20020)
    Destination port: 21100 (21100)
    Length: 110
    Checksum: 0x7f71 [validation disabled]
        [Good Checksum: False]
        [Bad Checksum: False]
MIH Protocol
    MIH Header
        .... 1... = Bit ACK Req: 1
        .... .0.. = Bit ACK Rsp: 0
        Message ID: MIH_Net_HO_Commit Request (0x3408)
            0011 .... = SID: Command Service  (3)
            .... 01.. = Opcode: Request (1)
            .... ..00 0000 1000 = AID: MIH_Net_HO_Commit (8)
        Payload Length: 94
    MIH Payload
        Source Identifier: 10.0.0.2 (10.0.0.2)
        Destination Identifier: 10.0.0.1 (10.0.0.1)
        Link Type: IEEE 802.11 (19)
        Target Network Info
            PoA Link Addr: Cisco-Li_a4:37:e4 (00:1c:10:a4:37:e4)
```

```
Assigned Resource Set
    QoS Resource Requirements
        Num Cos Types: 10
        Min Pk Tx Delay
            Cos ID: 1
            Packet delay/loss rate: 4113
        Avg Pk Tx Delay
            Cos ID: 2
            Packet delay/loss rate: 8225
        Max Pk Tx Delay
            Cos ID: 3
            Packet delay/loss rate: 12337
        Pk Delay Jitter
            Cos ID: 4
            Packet delay/loss rate: 16449
        Pk Loss Rate
            Cos ID: 5
            Packet delay/loss rate: 20561
Link action execution delay: 100
Link Actions List
    Link Type: IEEE 802.11 (19)
    MN Link Addr: IntelCor_63:5c:b9 (00:13:ce:63:5c:b9)
    Link Action
        Link AC Type: LINK_POWER_UP (4)
        Link AC ATTR
    Link action execution delay: 100
    Link Type: IEEE 802.11 (19)
    MN Link Addr: Netgear_e8:46:48 (00:09:5b:e8:46:48)
    Link Action
        Link AC Type: LINK_POWER_DOWN (3)
        Link AC ATTR
            .... ..1. = Link Res Retain: 0x01
    Link action execution delay: 200
```

6.2.3 Comparativa Router Advertisement- IEEE 802.21

En la siguiente figura (Figura 6.11) se representa la comparativa de los tiempos de handover utilizando IEEE 8021.21 (iniciado por la red) y utilizando router advertisement para las pruebas realizadas:

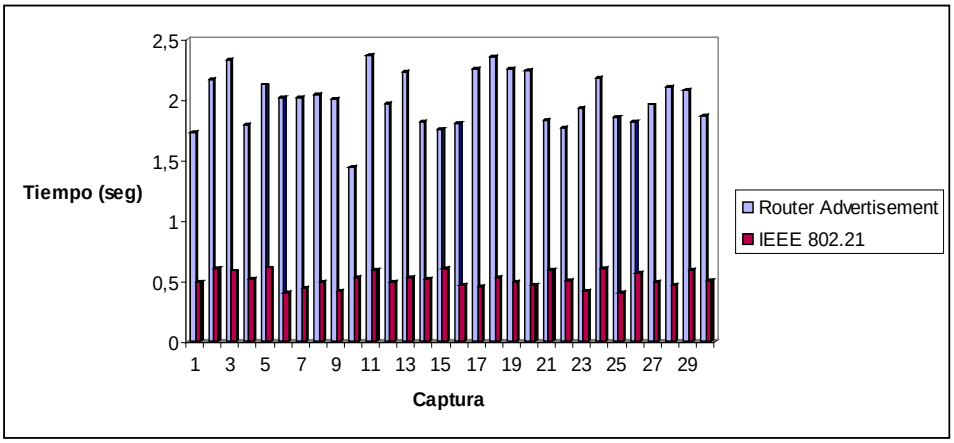

Figura 6.11: Comparativa IEEE 802.21 – Router Advertisement (Network Initiated)

Al igual que en el caso del handover iniciado por el MN el tiempo empleado para el handover usando IEEE 802.21 es bastante inferior al empleado usando router Advertisement ya que con el protocolo MIH se puede construir la CoA y obtener la dirección del router de acceso de la nueva red antes de realizar el handover mediante la información intercambiada con los mensajes de señalización IEEE 802.21. Por el contrario sin IEEE 802.21 es necesario recibir los router advertisement y a partir del prefijo indicado construir la CoA correspondiente.

A partir de la media de los tiempos empleados se observa que con la utilización del protocolo MIH se reduce aproximadamente un 75% el tiempo empleado en el handover, lo que supone una importante mejora.

7. Conclusiones

En este proyecto se han estudiado los mecanismos para el handover entre dos redes definidos en el estándar IEEE 802.21. Se han implementado las primitivas que utilizarán los usuarios del protocolo MIH así como los mensajes de señalización definidos por dicho protocolo que permiten el intercambio de información durante el handover ya sea iniciado éste por el Mobile Node o por la red.

Se han llevado a cabo pruebas experimentales que han consistido en la realización de handovers entre dos puntos de acceso Wifi pertenecientes a dos redes distintas para comprobar los beneficios de utilizar los mecanismos de handover del estándar IEEE 802.21.

En primer lugar se ha realizado la comparativa de los tiempos empleados para realizar el handover usando sólo los router advertisement proporcionados por el protocolo Mobile IPv6 frente a la utilización del protocolo MIH, cuando el handover es iniciado por el Mobile Node. En este caso analizando los resultados obtenidos para cada uno de estos casos se observa que la utilización de IEEE 802.21 proporciona una mejora sustancial ya que se reducen los tiempos de handover en un 75%. Esto es debido a que la información de la red donde se va a realizar el handover es conocida de antemano ya que los mensajes del protocolo MIH intercambian dicha información. Además se negocia, mediante este protocolo, la reserva de recursos en la red de destino así como se pone en funcionamiento el nuevo enlace para proporcionar conectividad tanto a nivel de enlace como a nivel IP.

En el segundo caso se ha realizado la comparativa de los tiempos empleados para realizar el handover usando sólo los router advertisement proporcionados por el protocolo Mobile IPv6 frente a la utilización del protocolo MIH, cuando el handover es iniciado por la red. Como en el caso anterior la reducción de tiempos del handover también es muy significativa (alrededor del 75%) por los mismos motivos explicados anteriormente.

Por lo tanto de los resultados experimentales se puede obtener la conclusión de que la utilización de las primitivas y mensajes definidos en el estándar proporcionan una optimización del proceso de handover ya que reducen los tiempos del mismo de una forma significativa lo que se traduce en una mejora de la calidad de servicio.

Los resultados experimentales obtenidos para la optimización del handover entre dos redes IEEE 802.11 pueden ser extrapolables a otro tipo de redes como IEEE 802.16.

La utilización de IEEE 802.21 proporciona un mecanismo de handover entre redes heterogéneas de forma estandarizada y que es posible utilizar entre los distintos tipos de redes existentes mejorando así la calidad de servicio en las redes de telecomunicaciones permitiendo a los usuarios moverse entre las zonas de cobertura proporcionadas por cada una de las redes manteniendo activa en todo momento la conexión existente pudiendo pasar de una red a otra en función de sus necesidades de una forma transparente.

Existen otras soluciones como UMA (Unlicensed Mobile Access) [26] pero estas sólo contemplan el handover entre redes de telefonía móvil (GSM/GPRS y UMTS) y redes IEEE 802.11 por lo que no proporcionan una solución global como la que proporciona IEEE 802.21, de ahí la conveniencia de usar el estándar IEEE 802.21.

8. Presupuesto

En este capítulo se van a evaluar los costes que podría suponer la realización de este proyecto y las descomposición de las tareas que serían necesarias para la realización del mismo representando éstas en un diagrama de Gantt [27].

8.1 Diagrama de Gantt

La descomposición de las tareas (y subtareas) sería la siguiente:

A. Documentación y análisis

A.1 Recopilación de documentación → 1 semana
A.2 Análisis de la documentación → 2 semanas
A.3 Búsqueda de soluciones → 1 semana

B. Desarrollo del software

B.1 Implementación de código → 12 semanas
B.2 Documentación del código → 4 semanas

C. Realización de pruebas del software

C.1 Definición de las pruebas → 1 semana
C.2 Montaje del escenario de las pruebas → 1 semana
C.3 Pruebas de funcionamiento → 3 semanas

Las tareas anteriores se representan en el siguiente diagrama de Gantt:

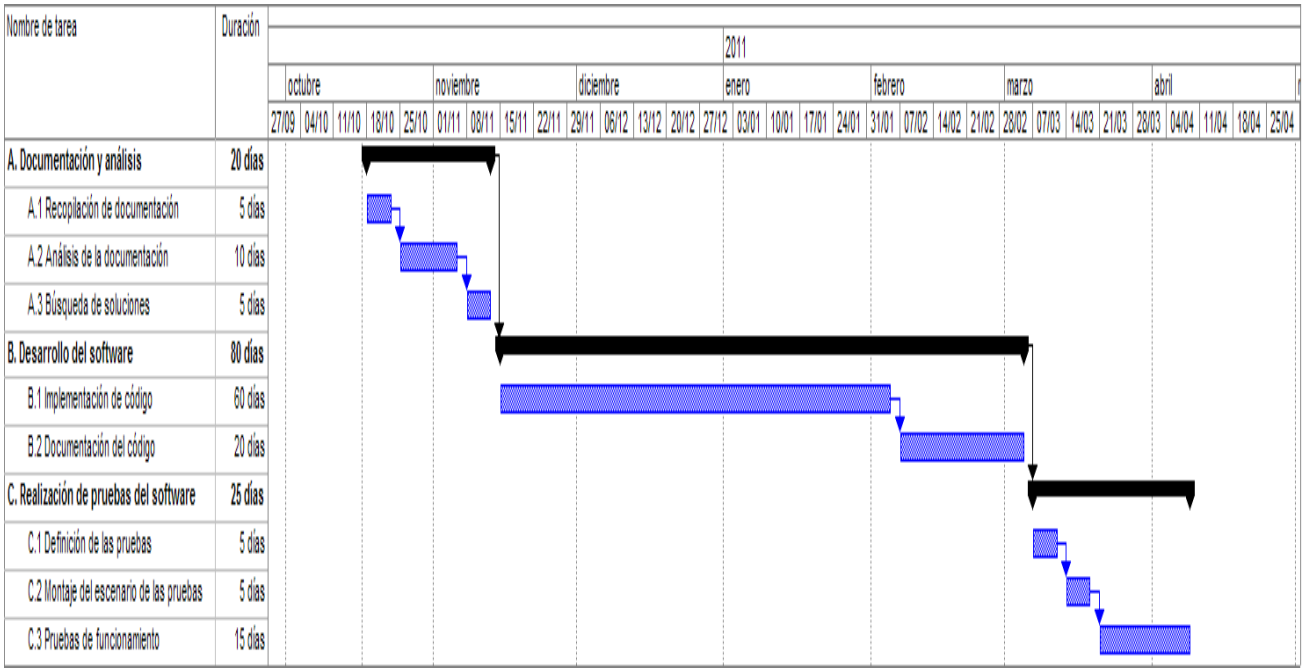

Figura 8.1: Diagrama de Gantt

8.2 Coste del material

En este coste se incluyen los medios que necesita el personal de trabajo para desarrollar el proyecto así como los alquileres de instalaciones y equipos para desarrollar las pruebas.

- Equipos informáticos: 2 portátiles valorados en 2000 €
- Alquiler de instalaciones y equipos: servidores, routers, equipos 2G/3G con un coste aproximado de 20000 €.
- Costes varios: fotocopias, impresión de documentos, etc... con un coste aproximado de 1000 €

8.3 Coste de personal

Aquí se tiene en cuenta el coste del personal encargado de realizar el proyecto, en este caso, se necesitan dos ingenieros para la realización del mismo, así como un director del proyecto. El cómputo se realiza a partir de las horas dedicadas a la realización del proyecto con un coste aproximado de 60 €/hora por cada ingeniero y de 100€/hora para el director de proyecto. Teniendo en cuenta los tiempos descritos en el apartado 8.1 el coste de los dos ingenieros sería el siguiente:

Fase	**Horas**	**Coste unitario (€)**	**Importe (€)**
Documentación y análisis	160	9600	19200
Desarrollo del software	640	38400	76800
Realización de pruebas del software	200	12000	24000
Total	**1000**	**60000**	**120000**

Tabla 8.1: Presupuesto Ingenieros de Proyecto

El coste del director del proyecto sería el siguiente:

Fase	**Horas**	**Importe (€)**
Documentación y análisis	8	8000
Desarrollo del software	32	32000
Realización de pruebas del software	10	10000
Total	**50**	**50000**

Tabla 8.2: Presupuesto Director de Proyecto

El coste total de personal resultaría:

Personal	**Importe(€)**
Ingenieros de Proyecto	120000
Director de Proyecto	50000
TOTAL	**170000**

Tabla 8.3: Presupuesto de Personal

8.4 Coste total

El coste total aproximado del proyecto sería la suma de los costes descritos en los apartados 8.2 y 8.3.

Concepto	**Importe (€)**
Coste del material	23000
Coste del personal	170000
TOTAL	**193000**

Tabla 8.4: Presupuesto Total

ANEXOS

ANEXO 1

Instalación del kernel de Linux para soporte Mobile IPv6

La versión del kernel de Linux utilizada en la realización de las pruebas es la 2.6.31, que se puede descargar desde http://www.kernel.org.

Los pasos a seguir para la instalación y compilación del kernel son los siguientes:

1.- Descomprimir el archivo linux-2.6.31.tar.bz2 en el directorio /usr/src:

- tar –xjvf linux-2.6.31.tar.bz2 –C /usr/src

2.- Acceder al directorio /usr/src/linux-2.6.31 y editar el archivo de configuración del kernel:

- cd /usr/src/linux-2.6.31
- make menuconfig

3.- Habilitar las especificaciones de movilidad para Mobile IPv6 activando las siguientes opciones:

- Code maturity level options
→ Prompt for development and/or incomplete code/drivers [CONFIG_EXPERIMIENTAL]

- General setup
→ System V IPC [CONFIG_SYSVIPC]

- Networking
→ Networking support [CONFIG_NET]
→ Networking options
  → Transformation user configuration interface [CONFIG_XFRM_USER]
  → Transformation sub policy support [CONFIG_XFRM_SUB_POLICY]
  → Transformation migrate database [CONFIG_XFRM_MIGRATE]
  → PF_KEY sockets [CONFIG_NET_KEY]
  → PF_KEY migrate [CONFIG_NET_KEY_MIGRATE]
  → TCP/IP networking [CONFIG_INET]
  → The IPv6 protocol [CONFIG_IPV6]
  → IPv6: AH Transformation [CONFIG_INET6_AH]

→ IPv6: ESP Transformation [CONFIG_INET6_ESP]
→ IPv6: IPComp Transformation [CONFIG_INET6_IPCOMP]
→ IPv6: Mobility [CONFIG_IPV6_MIP6]
→ IPv6: IPSec transport mode [CONFIG_INET6_XFRM_MODE_TRANSPORT]
→ IPv6: IPSec tunnel mode [CONFIG_INET6_XFRM_MODE_TUNNEL]
→ IPv6: MIPv6 route optimization mode [CONFIG_INET6_XFRM_MODE_ROUTEOPTIMIZATION]
→ IPv6: IPv6-in-IPv6 tunnel [CONFIG_IPV6_TUNNEL]
→ IPv6: Multiple Routing Tables [CONFIG_IPV6_MULTIPLE_TABLES]
→ IPv6: source address based routing [CONFIG_IPV6_SUBTREES]

- File System
→Pseudo filesystems
→ /proc file system support [CONFIG_PROC_FS]

4.- Compilar el kernel:

- make
- make modules
- make modules_install
- make install

5.- Crear una imagen initrd

- mkinitrd –o initrd.img-2.6.31 2.6.31

6.- Actualizar el gestor de arranque (GRUB)

- update-grub

ANEXO 2

Instalación del parche USAGI Mobile IPv6 para Linux

El parche está disponible en la página http:// umip.linux-ipv6.org. Una vez descargado el archivo, se descomprime y se siguen los siguientes pasos:

1.- Comprobar que el kernel soporta las extensiones para Mobile IPv6. Para ello se lanza el script chkconf_kernel.sh:

- ./chkconf_kernel.sh /usr/src/linux-2.6.31

2.- Desde el directorio principal donde se ha descomprimido el programa ejecutar el archivo de configuración habilitando el terminal virtual que permite depurar errores:

- ./configure --enable-vt

3.- Compilar e instalar el programa

- make
- make install

ANEXO 3

Instalación plugin del protocolo MIH para Wireshark

Debido a que el decodificador para los mensajes del protocolo MIH no está desarrollado en Wireshark, es necesario implementar un plugin que decodifique estos mensajes. Una vez implementado el plugin es necesario volver a compilar la aplicación para que esta tenga la nueva funcionalidad. Los pasos a seguir para compilar e instalar el nuevo plugin son los siguientes:

1.- Descargar el archivo con el código fuente de la aplicación desde la página de Wireshark: http:// www.wireshark.org

2.- Descomprimir el archivo

3.- Crear un nuevo directorio dentro del directorio plugins que va a contener los archivos necesarios para el funcionamiento del nuevo plugin. El nombre del directorio debe coincidir con el nombre del plugin, en este caso, mih, que es el nombre del protocolo que se ha desarrollado.

4.- El nuevo directorio (plugins/mih) debe contener los siguientes archivos:

- AUTHORS
- COPYING
- ChangeLog
- Makefile.am
- Makefile.common
- Makefile.nmake
- moduleinfo.h
- moduleinfo.nmake
- plugin.rc.in

Los ejemplos de estos archivos se encuentran en el directorio plugins/gryphon. Es necesario realizar una copia de estos archivos e instalarlos en el nuevo directorio (plugins/mih) para posteriormente modificarlos en función del nuevo protocolo.

Además el directorio plugins/mih debe contener el archivo del código fuente que se ha desarrollado para este nuevo protocolo:

- packet-mih.c

5.- Modificar los archivos AUTHORS, COPYING y ChangeLog introduciendo los datos relativos al desarrollador del nuevo plugin

6.- Modificar el archivo plugins/mih/Makefile.am. Para ello hay es necesario sustituir cada ocurrencia de “gryphon” por “mih”

7.- Modificar el archivo plugins/mih/Makefile.common. Hay que modificar el nombre del plugin y las fuentes, cambiando el existente, “gryphon”, por “mih”. Además hay que comentar la línea que hace referencia a los archivos de cabecera ya que no se utilizan en el desarrollo de este plugin.

8.- Mantener el archivo plugins/mih/Makefile.nmake sin realizar ningún cambio.

9.- Modificar el archivo plugins/mih/moduleinfo.h. Introducir la información de la versión relativa al nuevo protocolo.

10.- Modificar el archivo plugins/mih/moduleinfo.nmake. Introducir la nueva información en concordancia con los datos del archivo moduleinfo.h

11.- Mantener el archivo plugins/mih/plugin.rc.in sin realizar ninguna modificación.

12.- Modificar el archivo plugins/Makefile.nmake. Es necesario añadir una línea correspondiente el nuevo protocolo en la siguiente ubicación (según el orden alfabético):

- SUBDIRS = $ (_CUSTOMS_SUBDIRS_) \
  …
  gryphon \
  irda\
  …
  mih
  …

13.- Modificar el archivo plugins/Makefile.nmake. Es necesario añadir la siguiente secuencia de instrucciones (en el lugar que le corresponde según la ordenación alfabética del archivo):

- cd mih
  $ (MAKE) / $ (MAKEFLAGS) –f Makefile.nmake $ (PLUGIN_TARGET)
  cd ..

- xcopy plugins\mih*.dll $ (INSTALL_DIR)\plugins\$ (VERSION) /d
  cd plugins

14.- Modificar el archivo Makefile.am del directorio principal. Hay que añadir la siguiente instrucción en la siguiente ubicación (según el orden alfabético):

- if HAVE_PLUGINS
  plugin_ldadd = \
  …
  -dlopen plugins/gryphon/gryphon.la \
  -dlopen plugins/irda/irda.la \
  …
  **-dlopen plugins/mih/mih.la **
  ...

15.- Modificar el archivo configure.in del directorio principal. Hay que añadir la siguiente instrucción en la siguiente ubicación (según el orden alfabético):

- AC_OUTPUT (
  …
  plugins/gryphon/Makefile
  plugins/irda/Makefile
  …
  plugins/mih/Makefile
  …

16.- Modificar el archivo epan/Makefile.am. Hay que añadir la siguiente instrucción en la siguiente ubicación (según el orden alfabético):

- plugin_src = \
  …
  ../plugins/gryphon/packet-gryphon.c \
  ../plugins/irda/packet-irda.c \
  …
  **../plugins/mih/packet-mih.c **
  …

17.- Modificar el archivo packaging/nsis/Makefile.nmake. Añadir la siguiente instrucción (según el orden alfabético) en la lista de plugins para la variable PLUGINS de este archivo:

- ../../plugins/mih/mih.dll

18.- Modificar el archivo packaging/nsis/wireshark.nsi. Añadir la siguiente instrucción (según el orden alfabético) en la lista “File” en “Disector Plugins”de este archivo:

- File “..\..\plugins/mih/mih.dll “

19.- Ejecutar autogen.h

20.- Configurar e instalar la aplicación:

- ./configure --prefix=&(HOME)/build/root
- make install

Bibliografía

[1] “IEEE P802.21/D12.0: Draft Standard for Local and Metropolitan Area Networks: Media Independent Handover Services”. IEEE June 2008.

[2] Brian W. Kernighan, Dennis M. Ritchie. “The C programming language”. Second Edition. Prentice Hall

[3] Soliman, Hesham. “Mobile Ipv6: mobility in a wireless Internet”. Addison-Wesley 2004.

[4] Vivek Gupta. “IEEE P802.21 Tutorial”. July 2006.

[5] Antonio de la Oliva, Telemaco Melia, Albert Banchs, Ignacio Soto, Albert Vidal, Carlos J. Bernardos. “A case study: IEEE 802.21 enabled mobile terminals for optimized WLAN/3G handovers”. Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid, España.

[6] Antonio de la Oliva, Telemaco Melia, Albert Banchs, Ignacio Soto and Albert Vidal. “IEEE 802.21 (Media Independen Handover Service) Overview”. Departamento de Ingeniería Telemática, Universidad Carlos III de Madrid, España.

[7] Esa Piri and Kostas Pentikousis, The Internet Protocol Journal, Volume 12, No.2 IEEE802.21http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_12-2/122_ieee.html

[8] Steve Oualline. O’Reilly. “Practical C programming”. 3rd Edition.

[9] Sockets UDP en C para Linux. http://www.chuidiang.com/clinux/sockets/udp/udp.php, [online] Referencia visitada Junio 2010

[10] Apuntes de la asignatura Redes de Ordenadores. Cuarto Curso. Ingeniería de Telecomunicación. Universidad Carlos III de Madrid

[11] D. Jonson, C. Perkins, J. Arkko. “Mobility Support in IPv6”. RFC 3775. IETF 2004.

[12] Carlos J. Bernardos, Ignacio Soto and María Calderón, “IPv6 Network Mobility”, http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_102/102_ipv6.html.

[13] C. Perkins. “IP Mobility Support for IPv4”. RFC 3775. IETF 2002.

[14] Apuntes de la asignatura Redes y servicios de comunicaciones. Tercer Curso. Ingeniería de Telecomunicación. Universidad Carlos III de Madrid.

[15] Antonio de la Oliva Delgado. “Mecanismos de Control para Terminales Móviles en Entornos de Tecnologías de Acceso Heterogéneas”. Tesis Doctoral. Universidad Carlos II de Madrid. 2008.

[16] http://www.linksys.com. [online] Referencia visitada Noviembre 2009

[17] http://www.ubuntu.com [online] Referencia visitada Junio 2009

[18] http://www.kernel.org [online] Referencia visitada Junio 2009

[19] How to: Compile Linux kernel 2.6. http://www.cyberciti.biz/tips/compiling-linux-kernel-26.html . [online] Referencia visitada Junio 2009

[20] http://www.nautilus6.org. [online] Referencia visitada Julio 2009

[21] http://umip.linux-ipv6.org [online] Referencia visitada Octubre 2009

[22] http://www.wireshark.com. [online] Referencia visitada Septiembre 2009

[23] Linux IPv6 Router Advertisement Daemon (radvd). http://www.litech.org/radvd. [online] Referencia visitada Julio 2009

[24] Iperf. http://iperf.sourceforge.net. [online] Referencia visitada Octubre 2009

[25] Apuntes de la asignatura Estadística. Segundo Curso. Ingeniería de Telecomunicación. Universidad Carlos III de Madrid

[26] UMA Overview. http://www.umatechnology.org/overview. [online] Referencia visitada Septiembre 2010.

[27] Apuntes de la asignatura Proyectos de Ingeniería. Quinto Curso. Ingeniería de Telecomunicación. Universidad Carlos III de Madrid.